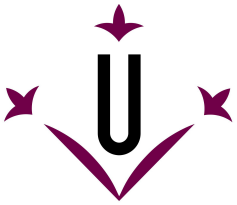

Universitat de Lleida
Escola Politècnica Superior
Màster en Interacció Persona-Ordinador

Treball de final de màster

Information Architecture for the Semantic Web

Autor: Josep Maria Brunetti Fernández
Director: Roberto García González

Setembre de 2012

Abstract

The proliferation of Linked Open Data and other data publishing initiatives on the Web has increased the amount of data available for analysis and reuse. The potential of this vast amount of data is enormous but in most cases it is very difficult for users to explore and use this data, especially for those without experience with Semantic Web technologies. Lay users find it difficult to explore and use Semantic Web Data due to the prevalence of specialised browsers that require complex queries to be formed and requiring intimate knowledge on how datasets are structured.

Our contribution to solving this problem is applying the data analysis mantra of "overview, zoom and filter". These interaction patterns are implemented using information architecture components users are already familiar with but that are automatically generated from data and ontologies.

This approach has been applied in Rhizomer, a tool capable of publishing Semantic Web datasets while facilitating user awareness of the published content. Rhizomer has been evaluated with end users as part of a User Centred Design development process. Iterative evaluations have motivated and guided the introduction of new features and validated requirements.

Contents

1 Introduction **11**
- 1.1 Motivation . . . 12
- 1.2 Contributions . . . 12
- 1.3 Outline . . . 13

2 State of the art **15**
- 2.1 The Semantic Web . . . 16
  - 2.1.1 RDF . . . 17
  - 2.1.2 RDF Schema . . . 18
  - 2.1.3 Ontologies and OWL . . . 19
  - 2.1.4 XML and XML Schema . . . 19
  - 2.1.5 SPARQL . . . 19
- 2.2 Open Data . . . 20
- 2.3 Linked Data . . . 20
- 2.4 Linked Open Data . . . 20
- 2.5 Related work . . . 22
  - 2.5.1 Linked Data browsers . . . 22
    - 2.5.1.1 Text-based browsers . . . 22
    - 2.5.1.2 Graph-based browsers . . . 23
    - 2.5.1.3 Summary . . . 24
    - 2.5.1.4 Rhizomer . . . 24

3 Preparation **27**
- 3.1 Approach . . . 28
  - 3.1.1 Problem . . . 28
    - 3.1.1.1 Semantic Web challenges . . . 28
  - 3.1.2 Hyphotesis . . . 28
    - 3.1.2.1 Information Architecture . . . 28
    - 3.1.2.2 Tasks for data analysis . . . 29
- 3.2 Methodology . . . 31
  - 3.2.1 MPIu+a . . . 31
    - 3.2.1.1 Overview . . . 31
    - 3.2.1.2 User-Centered Design . . . 32
      - 3.2.1.2.1 Usability . . . 32
      - 3.2.1.2.2 Accessibility . . . 32
    - 3.2.1.3 Software engineering . . . 33
      - 3.2.1.3.1 Requirements analysis . . . 33
      - 3.2.1.3.2 Design . . . 33

3.2.1.3.3 Implementation . . . 34
3.2.1.3.4 Launch . . . 34
3.2.1.4 Prototyping . . . 34
3.2.1.5 Evaluation . . . 35
3.2.2 RITE method . . . 36

4 Contribution **37**
4.1 Iteration 1 . . . 38
4.1.1 Requirements analysis . . . 38
4.1.1.1 Defining end users . . . 38
4.1.1.2 Requirements . . . 38
4.1.1.2.1 Functional requirements . . . 38
4.1.1.2.2 Non-functional requirements . . . 39
4.1.2 Design . . . 39
4.1.2.1 Navigation menus . . . 39
4.1.2.2 Facets . . . 40
4.1.2.3 Breadcrumbs . . . 41
4.1.2.3.1 Location breadcrumbs . . . 41
4.1.2.3.2 Path breadcrumbs . . . 41
4.1.2.3.3 Attribute breadcrumbs . . . 42
4.1.3 Prototyping . . . 42
4.1.4 Implementation . . . 42
4.1.4.1 Navigation menus . . . 42
4.1.4.2 Facets . . . 45
4.1.4.2.1 Automatic facet discovery and ranking . . . 46
4.1.4.3 Breadcrumbs . . . 48
4.1.5 Evaluation . . . 49
4.1.5.1 Objectives . . . 49
4.1.5.2 User profiles . . . 49
4.1.5.3 Methodology . . . 49
4.1.5.4 Tasks . . . 50
4.1.5.5 Usability metrics . . . 50
4.1.5.6 Results . . . 50
4.1.5.7 Conclusions . . . 52
4.2 Iteration 2 . . . 53
4.2.1 Requirements analysis . . . 53
4.2.2 Design . . . 53
4.2.3 Implementation . . . 55
4.2.4 Evaluation . . . 57
4.2.4.1 Tasks . . . 58
4.2.4.2 Usability metrics . . . 58
4.2.4.3 Results . . . 58
4.2.4.4 Conclusions . . . 59

5 Conclusion **61**
5.1 Conclusions . . . 62
5.2 Future work . . . 62

A User evaluation documents **65**

List of Figures

2.1 The Semantic Web layer cake 16
2.2 RDF graph describing Dr. Eric Miller 17
2.3 Linked Open Data (LOD) cloud 21
2.4 Rhizomer architecture overview 25

3.1 MPIu+a organization 31

4.1 Paper prototype 42
4.2 Software prototype 43
4.3 Generating a navigation submenu for DBPedia Species with 7 slots (left original, right result) 45
4.4 Navigation menu generated for the DBPedia 46
4.5 Automatic facets for the `http://dbpedia.org/ontology/Film` class 48
4.6 Pivoting enhancements 57

List of Tables

4.1 Evaluation results for the first iteration 51
4.2 Evaluation results for the second iteration: Efficiency (time on task) 58

List of Examples

1 (RDF/XML Syntax) 18
2 Structure of a SPARQL query 19
3 SPARQL query to get root classes 43
4 SPARQL query to get direct subclasses for the given class 43
5 SPARQL query to get the number of instances of each instantiated class . . 43
6 Overview of the navigation menu generation algorithm 44
7 SPARQL query to obtain all properties for a <CLASS> 47
8 SPARQL query to obtain values and counts for a <CLASS> and <PROPERTY> 47
9 SPARQL query to obtain the number of instances of a <CLASS> that have a concrete <PROPERTY> 47
10 SPARQL query to obtain the cardinality for a concrete <CLASS> and <PROPERTY> 48
11 SPARQL query that retrieves at most the 5 most common classes instantiated by the values of a facet 56
12 SPARQL query that computes the most specific common superclass 56
13 Generated SPARQL query before pivoting 56
14 Generated SPARQL query after pivoting 57

Chapter 1

Introduction

1.1 Motivation

It has been a long time since Tim Berners-Lee invented the World Wide Web (WWW) [1] and during this time the Web has evolved and has become more sophisticated. Its huge success present new requirements. Nowadays, the factors to consider when creating a website are not only technological but also of information structure, contents and functionalities.

The amount of data available in the Web, in its transition to a Web of Data or Semantic Web, is increasing at an astounding rate. Despite the Semantic Web was proposed more than ten years ago [2], it hasn't been until recently when it has started to become popular, especially thanks to Linked Open Data initiatives like those conducted by the USA, UK or Spanish government agencies towards a greater level of transparency.

The objective of this initiative is to motivate the publication of Open Data in formats that are more easily integrable, queryable and that facilitate its reuse. The cloud of interrelated and open datasets included in the LOD cloud has rapidly evolved, from the 2 billion statements and 30 datasets one year after its creation in February 2007, to more than 31 billion statements[1] and 295 datasets in September 2012.

The potential of this vast amount of data is enormous but in most cases it is very difficult and cumbersome for users to visualize, explore and use this data, especially for lay-users without experience with Semantic Web technologies. From the end-user perspective, the available datasets are monolithic and opaque files, which usually can just be explored using complex semantic queries or complex user interfaces.

Visualizing and interacting with Linked Data is an issue that has been recognized from the beginning of the Semantic Web (cf. e.g. [3]). However, the Semantic Web has not yet been adopted by end users [4]. This is due in part to the fact that users find it difficult to use. Sometimes even advanced users of the Semantic Web find it complicated [5]. Existing tools make it difficult for users to explore a dataset, most of them require technical skills and in most cases the results are not very usable. The transition of the web to the web of data requires the support for a rich user interaction.

The objective is now to try to make all this data more usable so users that are not Semantic Web experts, when facing a dataset, can easily grasp what kind of entities are contained therein, how they are interrelated, what are the main properties and values, etc. This will increase the awareness of the semantic data currently available on the Web and also facilitate the development of new and innovative applications on top of it. The overall outcome will be that available data increases its impact and the society as a whole benefits more from semantic data.

1.2 Contributions

The main contributions of this work are:

- The identification of the most common end-user tasks when interacting with semantic web data.
- The proposal of a set of interaction patterns to perform these tasks and the information architecture components to implement each pattern.

[1] `http://www4.wiwiss.fu-berlin.de/lodcloud/state/`

- A comprehensive, generic and scalable implementation of the information architecture components identified. These components allow lay-users to obtain an overview of semantic datasets and explore them. Overall, they improve the usability and accessibility of semantic data.

1.3 Outline

The complete outline of this document is as follows. Chapter 2 provides the background necessary to contextualize this work: the Semantic Web, its core technologies and the Linked and Open Data initiatives.

Chapter 3 identifies the main challenges when interacting with the Semantic Web and the hypotheses to solve them. It also introduces the methodology followed in this work.

Chapter 4 is the core part of this work, featuring the whole development process of the Information Architecture components based on Semantic Web technologies. The development is divided into different iterations.

Chapter 5 concludes this work with the conclusions and future work.

Finally, Appendix A includes the documents used in the user tests: confidentiality document, post-task questionnaire and post-test questionnaire.

Chapter 2

State of the art

2.1 The Semantic Web

Tim Berners-Lee's vision of a Semantic Web is almost as old as the web itself. However, it took a few more years to be defined, starting 1999, when he wrote his book "Weaving the Web" [6], where he described his dream for the Web and introduced the *Semantic Web*:

> *I have a dream for the Web... and it has two parts.*
>
> *In the first part, the Web becomes a much more powerful means for collaboration between people. I have always imagined the information space as something to which everyone has immediate and intuitive access, and not just to browse, but to create [...]*
>
> *In the second part of the dream, collaborations extend to computers. Machines become capable of analyzing all the data on the Web - the content, links, and transactions between people and computers. A "Semantic Web", which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy, and our daily lives will be handled by machines talking to machines, leaving humans to provide the inspiration and intuition [...]*

Some years later, in 2001, Tim Berners-Lee described in more detail the Semantic Web [2] as an extension of the current Web, with contents and aimed not only for humans but also for computer agents. The word semantic itself implies meaning or understanding. In the Semantic Web, information is given well-defined meaning, better enabling computers and people to work in cooperation.

During this time, the W3C published a set of standards, markup languages and official recommendations related with the Semantic Web. They are summarized in Figure 2.1.

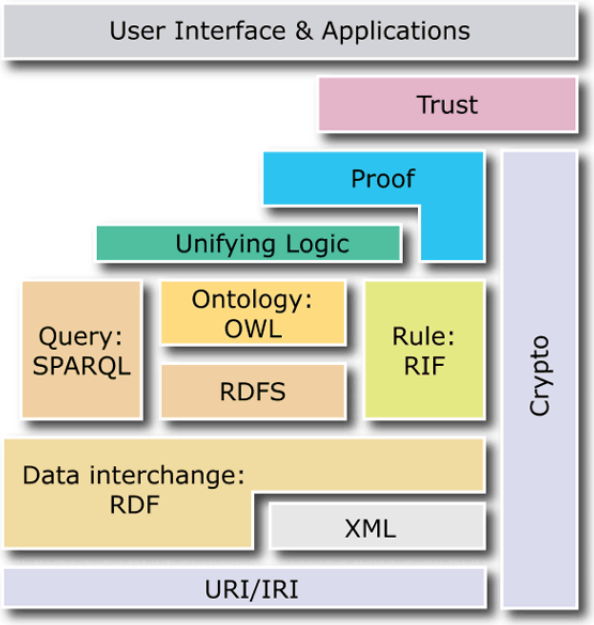

Figure 2.1: The Semantic Web layer cake

Nowadays, the Semantic Web is a collaborative movement led by the World Wide Web Consortium (W3C). According to the W3C [7], *"The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries"*.

2.1.1 RDF

The Resource Description Framework (RDF) [8] is a standard model for data interchange on the Web. It is the W3C standard for identifying resources and expressing statements and about them. It allows data to be mixed, exposed and shared across different applications. Using this model, data cannot only be viewed by humans, but also consumed and processed by applications.

RDF is based upon the idea of making statements about resources. Statements are stored as subject-predicate-object expressions, which are known as triples. In each triple, the subject denotes a resource and it is represented by an URI, which makes it globally identifiable. The predicate denotes an aspect of the resource and expresses a relationship between the subject and the object. The object is either a resource or a literal. The linking statements structure forms a labeled directed graph, composed by nodes and directed edges between nodes. The edges represent the named link (the predicate or property) between two resources, represented by graph nodes.

Figure 2.2, taken from the W3C website, describes "Eric Miller": "there is a Person identified by http://www.w3.org/People/EM/contact#me, whose name is Eric Miller, whose email address is em@w3.org, and whose title is Dr.".

Figure 2.2: RDF graph describing Dr. Eric Miller

The graph representation is often used to visually understand RDF. Apart from this representation, RDF can be stored in different formats. The most common serialization format is XML [9]:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
             xmlns:contact="http://www.w3.org/2000/10/swap/pim/contact#">

  <contact:Person rdf:about="http://www.w3.org/People/EM/contact#me">
    <contact:fullName>Eric Miller</contact:fullName>
    <contact:mailbox rdf:resource="mailto:em@w3.org"/>
    <contact:personalTitle>Dr.</contact:personalTitle>
  </contact:Person>

</rdf:RDF>
```

Example 1: (RDF/XML Syntax)

2.1.2 RDF Schema

RDF Schema (RDFS) [10] is a semantic extension of RDF designed to create RDF vocabularies. It provides the basic elements to describe ontologies (more details in Section 2.1.3. These are the more relevant primitives of RDF and RDFS:

- `rdfs:Resource:` is the class of everyhing. All things described by RDF are instances of the class `rdfs:Resource` and all other classes are subclasses of this class.
- `rdfs:Class`: declares a resource as a class. A class models a concept with some characteristics. The definition of `rdfs:Class` is recursive: `rdfs:Class` is an instance of `rdfs:Class`.
- `rdfs:Literal`: is the class of literal values such as strings and integers. Literals can be plain or typed. `rdfs:Literal` is an instance of `rdfs:Class` and a subclass of `rdfs:Resource`.
- `rdfs:DataType`: is the class of datatypes. All instances of `rdfs:Datatype` are a subclass of `rdfs:Literal`.
- `rdf:Property`: is the class of RDF properties. `rdf:Property` is an instance of `rdfs:Class`.
- `rdf:type`: is an instance of `rdf:Property` used to state that a resource is an instance of a class.
- `rdfs:label` is an instance of `rdf:Property` that can be used to provide a human-readable name of a resource.
- `rdfs:subClassOf` is an instance of `rdf:Property`. The triple `C1 rdfs:subClassOf C2` states that the class C1 is a subclass of class C2. The `rdfs:subClassOf` property is transitive.
- `rdfs:subPropertyOf` is an instance of `rdf:Property`. The triple `P1 rdfs:subPropertyOf P2` states that the property P1 is a subproperty of property P2. The `rdfs:subPropertyOf` property is transitive.
- `rdfs:range`: is an instance of `rdf:Property`. It is used to state that the values of a property are instances of one or more classes.
- `rdfs:domain` is an instance of `rdf:Property`. It is used to state that any resource that has a given property is an instance of one or more classes.

By using these elements it is possible to define classes, hierarchies of classes, properties and hierarchies of properties.

2.1.3 Ontologies and OWL

In computer science, ontologies represent knowledge of a shared conceptualization [11]. An ontology captures and formalises objects or concepts within a domain, their properties and relationships among those concepts.

The Web Ontology Language (OWL) [12] is a semantic markup language for authoring and sharing ontologies on the Web. OWL is developed as a vocabulary extension of RDF and it is based on description logics, making it posible to reason about the entities within that domain.

2.1.4 XML and XML Schema

The Extensible Markup Language (XML) [13] is a markup language to encode documents so information can be more easily shared. The design goals of XML emphasize simplicity, generality, and usability over the Internet applications. Many applications have been developed to process XML data since it allows freedom in structure.

XML Schema describes the structure of XML documents. It can be used to define rules and restrictions to which an XML document must conform in order to be considered valid according to that schema. XML Schema provides simple and complex data types such as dates, numbers, strings, etc.

2.1.5 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) [14] is an RDF query language to retrieve and manipulate data stored in RDF. It is considered as one of the key technologies of the Semantic Web and it is has become an official W3C recommendation.

A SPARQL query comprises, in this order:

1. **Prefix declarations:** for abbreviating URIs.
2. **Dataset definition:** stating what RDF graph(s) are being queried
3. **A result clause:** identifying what information to return from the query
4. **The query pattern:** specifying what to query for in the underlying dataset
5. **Query modifiers:** ordering, slicing and otherwise rearranging query results

```
1  # prefix declarations
2  PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
3  ...
4  # dataset definition
5  FROM ...
6  # result clause
7  SELECT ...
8  # query pattern
9  WHERE {
10     ...
11 }
12 # query modifiers
13 ORDER BY ...
```

Example 2: Structure of a SPARQL query

2.2 Open Data

Open data is the idea of making data freely available to everyone to use. Data is shared without restrictions from patents or copyright. Open data has gained popularity with the rise of the World Wide Web and with the launch of open data government initiatives.

2.3 Linked Data

Linked Data [15] is a W3C movement about using the Web to connect datasets. Linked Data describes methods and a set of best practices for publishing and connecting structured data on the Web. It can be viewed as a subset of the Semantic Web movement.

Linked Data is builded upon two standard Web technologies: HTTP and URIs. Entities are identified by URIs and they can be looked up simply by dereferencing the URI over the HTTP protocol. The HTTP protocol provides a mechanism for retrieving resources. URIs and HTTP are supplemented by RDF, which provides a generic data model to describe resources.

Linked Data is based on four principles [16]:

- Use URIs as names for things.
- Use HTTP URIs so that people can look up those names.
- When someone looks up a URI, provide useful information, using the standards.
- Include links to other URIs. so that they can discover more things.

2.4 Linked Open Data

The Linked Open Data (LOD) community project aims to extend the Web by publishing Open Data using Linked Data principles. Nowadays, the LOD has grown to 295 datasets, 31 billion RDF triples, interlinked by around 504 million RDF links.

Figure 2.3 show the LOD cloud. Each node in the diagram represents a distinct dataset published as Linked Data. The arcs indicate that RDF links exists between resources in the two connected datasets.

Linked Open Data has become popular especially thanks to initiatives conducted by governments such as United Kingdom[1] or USA[2]. They use LOD for making distributed information publicly available [17].

[1] `http://data.gov.uk`
[2] `http://data.gov`

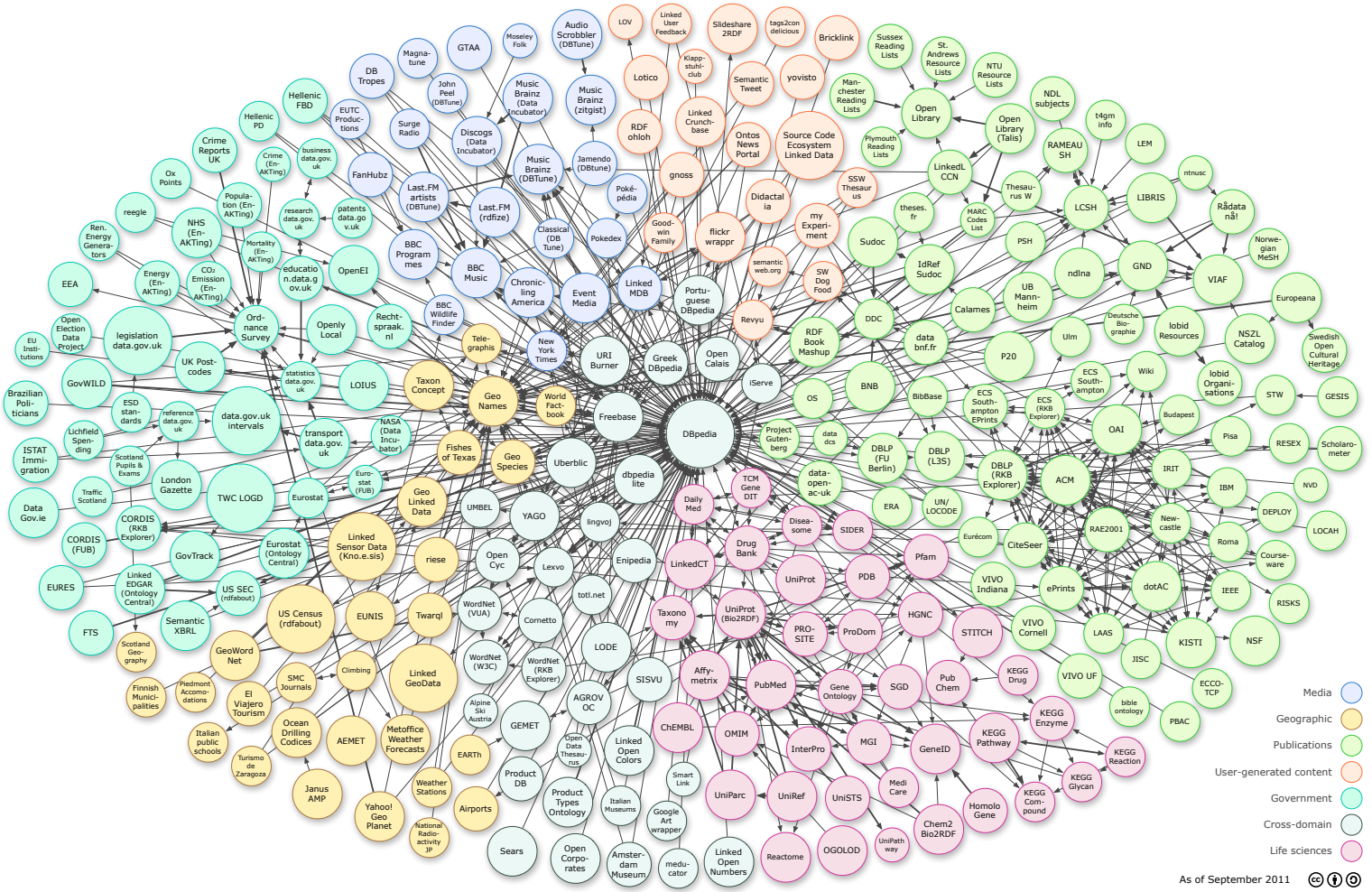

Figure 2.3: Linked Open Data (LOD) cloud

2.5 Related work

Publishing and presenting Linked Data in an accessible way for users has been addressed by several projects and different kind of tools. In this chapter we present existing initiatives that are related with this work. It is organized in different categories.

2.5.1 Linked Data browsers

The first tool that comes to mind when trying to realise what a dataset is about is a browser. Semantic Web browsers differ from Web browsers because they are not prepared for navigating documents but triples, the fundamental building block of the Web of Linked Data.

2.5.1.1 Text-based browsers

Disco
The *Disco - Hyperdata Browser* [18] is a simple browser for navigating Semantic Web resources. It renders as an HTML page all the information that can find about a specific resource. Users must provide the URI of a concrete resource to start navigation. Retrieved information is displayed as a property-value table. *Disco* also renders hyperlinks that allow users to navigate between related resources.

Tabulator
Tabulator [19, 20] is a generic RDF browser and editor. In addition to the rendering of properties and values for a resource, Tabulator provides specialised visualisations like maps for geo-located resources or timelines for time-framed ones.

Explorator
Explorator [21] is a tool that makes it possible to browse a dataset available through a semantic queries service. Though Explorator makes it possible to browse the dataset by combining search, facets or operations on sets of resources, it makes it also difficult to get a broader view on the dataset other than a list of all the classes or properties used. Its interface is difficult to understand for Lay-users.

Longwell
Longwell [22] is a tool part of the *Simile Project*, which provides a graphical user interface for generic RDF data exploration in a web browser. It provides a faceted browser that allows users to search large models by filtering through properties and values. *Longwell* shows a list of the currently filtered resources (RDF subjects) in the main part of the screen and a list of filters in the side. Each filter corresponds to a property of the resources with its values and their frequency. It can be configured to choose and prioritize which facets should be shown when the page loads, or it can choose heuristically which are the most important and should be selected.

/facet
/facet [23] is a generic browser for heterogeneous semantic web repositories. Users can select and navigate facets of resources of any type. It provides also a time-related facet visualization and can display geographical information on yahoo maps.

DBpedia Faceted Browser
The *DBpedia Faceted Browser* [24] is a project from Neofonie which allows users to make complex queries against DBpedia [25]. It supports keyword queries and offers relevant facets to filter search results, based on the DBpedia Ontology. The *DBpedia Faceted Browser* is a useful tool for browsing the DBpedia but it is not a generic browser, it only works for this concrete dataset.

Virtuoso OpenLink Data Explorer
Also known as *OpenLink RDF Browser* [26], *Virtuoso OpenLink Data Explorer* is a web browser for interacting with Linked Data that provides a basic faceted view. It requires an entity URI as input or a text string to look for. Consequently, the facets view is limited to the resources retrieved from a previous search. Moreover there is no way to previously get an overview of the kinds of resources in the dataset.

Marbles
Marbles [27] is a text-based RDF browser that retrieves information about resources by querying Semantic Web indexes and search engines. It displays resource information presented as property-value pairs in a table. It is necessary to provide a URI as input or it can be used also as a SPARQL endpoint.

BrowseRDF
BrowseRDF [28] is a faceted interface for arbitrary RDF data. Users can browse a dataset by constraining one or several of the facets using different operators: basic selection, join selection, inverse selection, etc. The authors also propose three metrics to rank facets automatically and choose those that best navigate the dataset.

2.5.1.2 Graph-based browsers

RDF-gravity
RDF Gravity [29] is a tool for visualizing RDF/OWL ontologies. It provides a graph visualization including different node shapes and edge decorations to distinguish different resource types. The tool allows users to specify filters to have specifiec views on the graph. It is also posible to perform a text search and SPARQL queries over classes, properties and instances.

IsaViz
IsaViz [30] is an interactive RDF graph browser and editor. It provides a 2.5D user interface that allows to zoom and navigate through the graph. *IsaViz* can render RDF graphs using Graph Stylesheets (GSS) [31], a stylesheet language derived from CSS and SVG for styling RDF models represented as node-link diagrams. It also supports the use of Fresnel lenses [32] to display resources of interest.

Fenfire
Fenfire [33] is a generic RDF browser and editor. The user interface is a graph visualization of the RDF data model. To allow scalability to large amouts of data, it displays only a central node and its neighbours. It is necessarty to provide a URI to start navigating.

2.5.1.3 Summary

Dadzie and Rowe [34] present the most exhaustive and comprehensive survey to date of existing approaches to visualising and exploring Semantic Web data, particularly Linked Data. They conclude that most of the tools are designed only for tech-users and do not provide overviews on the data. Browsers are especially useful when dealing with a dataset published as Linked Data because they provide a smooth browsing experience through the graph, e.g. *Disco* [18] or *Tabulator* [19, 20]. However, most of them do not provide additional support for getting a broader view of the dataset being browsed, just a view on the current resource.

Other tools like *Explorator* [21] or *Marbles* [27] allow to browse a dataset available through a semantic queries service. *Explorator* also makes it possible to browse the dataset by combining search, facets or operations on sets of resources, it is still difficult to get a broader view on the dataset other than a list of all the classes or properties used. In some cases it is also possible to get more informative components like facets, e.g. */facet* [23], *BrowseRDF* [28] or emphVirtuoso OpenLink Data Explorer [26] . However, in some cases, facets are pre-computed and just available for a given dataset as in the case of the *DBPedia Faceted Browser* [24].

Graph-based tools such as *Fenfire* [33], *RDF-Gravity* [29] or *IsaViz* [30] provide node-link visualizations of the datasets and the relationships between them. Although this approach can help obtaining a better understanding of the data structure, in some cases graph visualization does not scale well to large datasets [35]. Sometimes the result is a complex graph difficult to manage and understand [36].

To summarize, most of the existing tools make it difficult for non-technical users to explore linked data or they are restricted to concrete domains. None of them provide generic visualizations for RDF data. Consequently, it is still difficult for end users to obtain an overview of datasets, comprehend what kind of structures and resources are available and what properties resources typically have and how they are mostly related with each other.

2.5.1.4 Rhizomer

The survey [34] presented in the previous section is used to situate our contribution, implemented in a tool called Rhizomer and available online[3]. Rhizomer can be classified mainly in the category of text-based visualisation tools, though it also includes graphical representations for dataset overviews. However, it is important to note that it is not intended as a Linked Data browser. It is geared towards publishing a dataset and generating the user interface to improve user interaction for that specific dataset.

First of all, Rhizomer is based on a simple architecture, which makes it flexible, scalable and capable of adapting to different deployment and use scenarios. Its core is rooted on simple HTTP mechanisms and follows a REST approach [37]. Rhizomer also implements content negotiation taking into account the requested content type thus providing the requested data in the desired format.

Each resource is managed through the URI referencing where it is published, thus basing the whole system on a Resource Oriented Approach. The basic HTTP commands allow managing each resource: GET retrieves the semantic data associated with the resource in the requested format, PUT updates the data for the resource with the

[3] `http://rhizomik.net/rhizomer/`

submitted one, POST creates a new resource with the submitted semantic description and DELETE removes the specified resource and the corresponding data.

All the previous HTTP commands are forwarded to the underlying data store, see Figure 2.4. Currently, Rhizomer integrates connectors for Jena and Virtuoso. These connectors make it possible to implement all the data management operations.

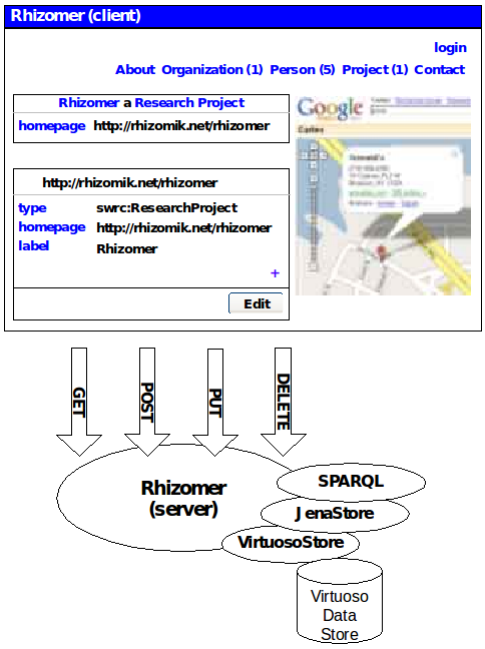

Figure 2.4: Rhizomer architecture overview

The client-side functionalities have been developed with the aim of improving the usability of the user interface. They are deployed in the user's browser and implemented using JavaScript and asynchronous HTTP calls (AJAX [38]), thought most of the functionality is also available without JavaScript in order to improve accessibility [39].

Like many Semantic Web browsers of data publishing tools, Rhizomer provides an HTML view on the data that also facilitates the navigation across the data graph. The RDF syntax of semantic data is completely hidden in order to increase usability. However, as it has been shown in the related work section, this approach does not contribute towards an awareness of the overall structure of a dataset.

More details about Rhizomer's implementation and functionalities can be found in [40].

Chapter 3

Preparation

3.1 Approach

3.1.1 Problem

3.1.1.1 Semantic Web challenges

The first thing to take into account when considering user interaction in the Semantic Web is that despite the fact that it was designed for machine consumption, at the end, humans are its real consumers. Therefore, end users should have usable tools and simple methods to explore the Web of Data. For such interaction to occur, some key usability challenges in using Linked Data have been identified [41, 42, 43, 44, 45]:

- **Exploration starting point:** most of existing LD browsers assume the end user will start browsing from a specific URI. However, most of end users don't even know what a URI means. They need an exploration starting point.
- **Combating information overload:** presenting all the properties and relations of a given resource can lead to information saturation. All these information should be presented in a more legible form to lower the cognitive load [46].
- **Getting an overview:** as data set size increases, human ability to obtain a good mental overview and retain information in memory decreases. This poses a challenge for large amounts of complex and heterogeneous data [47, 48].
- **Returning something useful:** RDF is the standard for resource descriptions but mainstream users don't understand it. It is necessary to present data in a usable way for users and hide its complexity.
- **User interaction:** users are familiar with the traditional Web and its browsable nature. The user interaction should be replicated and adapted to the Semantic Web and Linked Data.
- **Scalability:** when facing the Web of Data, scalability becomes very important. Applications based on Linked Data should be able to handle large datasets.
- **Generic interfaces:** in most cases, the Semantic Web has no immediate access [49] and it is only accessible through user interfaces for specific domains.

3.1.2 Hyphotesis

If non-technical end users are to use the Web of Data, they must employ tools that allow them to do so, focusing on the user interface and usability. To achieve this challenge, the proposal is to explore the typical tasks for data analysis and to draw from the experience in the Information Architecture domain, adapting them to the context of the Semantic Web.

3.1.2.1 Information Architecture

Information Architecture (IA) is the art and science of organizing information. In the context of the World Wide Web, Information Architecture [50] is the discipline that organizes and labels the information on websites. It englobes analyzing the contents, organizing web pages and designing the navigation systems.

The challenge of structuring and presenting semantic data to end users can be addressed with the experience accumulated in the Information Architecture domain. Information Architecture focuses its efforts in this problem, especially in complex systems and situations with great amounts of information. A good IA can improve the quality of a website and users can find more easily the information they are looking for.

Information Architecture identifies four kinds of systems:

- **Organisation systems:** they present information in different ways, following different schemas that make it possible to group or differentiate information using different criteria, like chronological or alphabetic order.
- **Navigation systems:** they help users move across the available information. For instance, there are navigation bars or site maps.
- **Labelling systems:** they describe categories, options and links using terms that are meaningful for users. They are all around the information architecture of a site, even as part of other systems, e.g. navigation bars labels.
- **Search systems:** they allow users to search specific information based on some sort of keywords. They also offer mechanisms to restrict the search space.

The drawback of all these IA systems is that they are quite expensive to develop and maintain. Nowadays, when developing a website, the procedure usually begins by defining the Information Architecture for the domain of this site with the help of the future users of the website. The obtained Information Architecture is usually based on formalisms that allow to represent only a small part of the domain semantics. Therefore, the process of creating a website from this type of Information Architecture is a heavy process that requires a lot of time and effort by developers, mainly because little automatization can be accomplished.

Fortunately, when these IA systems are built on top of the highly structured data typical in the Semantic Web and Linked Data, it is possible to automate most of the development and maintenance work. The Semantic Web provides methods and tools to model the information architecture of a concrete domain with more detail and in a formal way. This allows the use of automatic tools to process it in a more sophisticated way.

This work focuses on developing generic and automatic navigation systems for the Semantic Web. The objective is to reuse and adapt existing IA components that allow users to navigate through data.

3.1.2.2 Tasks for data analysis

As the volume of information in the Semantic Web increases, interacting with information becomes a more difficult task. To interact with these amounts of data, users use different ways or strategies depending on their goals.

For data analysis and information visualization, Shneiderman [51] proposed a set of tasks based on the visual information seeking mantra: “overview first, zoom and filter, then details on demand”. The starting point is the fundamental set of tasks for data analysis proposed by Schneiderman. We have explored the most appropriate Interaction Patterns to perform these tasks and the Information Architecture components to implement each pattern:

- **Overview:** to get a full view of the entire collection. We propose to apply the Global Navigation interaction pattern[1] or the Directory Navigation pattern[2]. In the context of IA the main components that support this pattern are navigation menus and site maps.
- **Filter:** to select items of interest and filter out uninteresting items through exploratory search. The proposal is the Faceted Navigation pattern[3], which in the context of IA corresponds to facets.
- **Details:** after filtering resources, the user can view detailed information about of them. The proposal is the Details on Demand pattern[4]. In the context of IA and Linked Data it corresponds to a HTML view with the properties and values of the resource of interest or a specific visualization, e.g. a map for geolocated resources.
- **History:** to keep a history of actions and support undo and replay actions. We propose to apply the Breadcrumb Navigation pattern[5]. In the context of IA it corresponds to breadcrumbs.

The proposal is to elaborate these interaction patterns and IA components in the context of the Semantic Web and Linked Data. These patterns and IA components have been chosen because they are simple, very common in websites and they have become essential about how to structure the information in the Web.

[1] http://www.welie.com/patterns/showPattern.php?patternID=main-navigation
[2] http://www.welie.com/patterns/showPattern.php?patternID=directory
[3] http://www.welie.com/patterns/showPattern.php?patternID=faceted-navigation
[4] http://www.welie.com/patterns/showPattern.php?patternID=details-on-demand
[5] http://www.welie.com/patterns/showPattern.php?patternID=crumbs

3.2 Methodology

3.2.1 MPIu+a

The methodology followed in this project is based on the MPIu+a development process [52]. MPIu+a is a development framework for interactive systems that integrates the discipline of Software Engineering with the basis of Human-Computer Interaction, Usability and Accessibility.

3.2.1.1 Overview

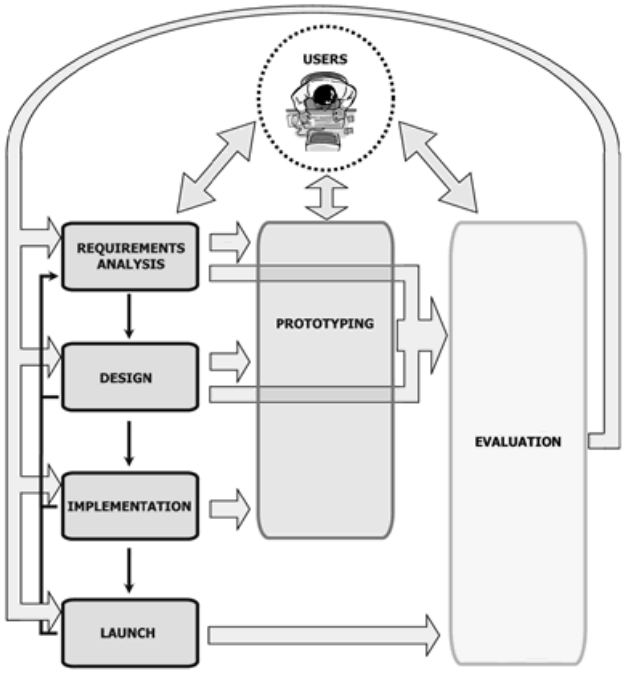

Figure 3.1: MPIu+a organization

Figure 3.1 shows the process model schema with its phases and the relation between them. These are the main features of this process model:

- **Conceptual organization:** the schema is organised in modules or stages that show the current phase.
- **Three main pillars:** the schema shows these three main pilars in different colors:
  - Software engineering: the classic software development life cycle based on the waterfall model (left column).
  - Prototyping: grouping all the techniques to build software samples to facilitate the subsequent evaluation (center column).
  - Evaluation: covering all the usability and accessibility validation methods.
- **The user:** a User Centred Process Model has the user as the most important part. This scheme reflects this meaning at the first glance, placing the User in central and above the other phases.

- **An iterative model:** the schema has a series of arrows to show the relation between phases and users active participation in some of them: requirements analysis, prototyping and evaluation.

3.2.1.2 User-Centered Design

User-Centered Design (UCD) is an type of design process for interactive systems focused on the users who will use the system. In UCD the user participates in all the stages of the design process. User requirements are considered from the beginning and included into the whole development cycle. These requirements are defined and refined through different methods such as ethnographic studies, focus groups, usability testing, etc.

It is important to note that User-Centered means focusing on all users. This implies considering all their differential characteristics and also thinking about those with a disability [53].

The ISO 13407 [54] standard establishes a framework that provides guidance to achieve the development of usable interactive systems incorporating the UCD during the development life cycle. This standard describes the User-Centered Design as a multidisciplinary activity that also includes human factors and ergonomic techniques.

3.2.1.2.1 Usability

The concept of usability is defined as the ease of use and learnability of a human-made object. Usability is a property that can be applied not only to software systems, but also to elements of our everyday life [55].

In software systems, the concept of usability was introduced by J. Nielsen [56] as a quality attribute that assesses how easy user interfaces are to use. The ISO 9241-11 [57] defines usability as "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use."

Usability is defined by 5 quality components:

- **Learnability:** how easy is it for users to accomplish basic tasks the first time they use a system.
- **Efficiency:** how quickly can users perform tasks once they have learned the system's design.
- **Memorability:** how easily can users reestablish proficiency when they return to a system after a period of not using it.
- **Errors:** how many errors do users make, how severe are them, and how easily can they recover from them.
- **Satisfaction:** how pleasant is it to use the system.

3.2.1.2.2 Accessibility

Accessibility is the degree to which a product or service is available to as many people as possible. Nowadays, providing accessibility has become an important factor in interactive

systems. The ISO standard [58] provides a guide with the ergonomic specifications to design computer interfaces.

Web accessibility [59] means that everyone can use the web, no matter their possible disabilities. The concept of web accessibility is often used to focus on people with disabilities or special needs. However, web accessibility can benefit everyone; old people or people with temporary disabilities (vision problems, injuries, etc.) can also benefit from it. Web accessibility can also improve the user interaction. The improvements in usability, navigation and content structure benefit everyone.

3.2.1.3 Software engineering

3.2.1.3.1 Requirements analysis

Requirements analysis in software engineering covers all the tasks to determine the user needs and conditions regarding to a concrete software or product. This phase is very important in every development and necessary to obtain good final results. This way, the number of errors is reduced because most factors were already considered in this phase.

Requirements analysis can be divided into the following activities [60]:

- Eliciting requirements: collecting the requirements from users and other stakeholders using different techniques.
- Analyzing requirements: determine whether the requirements are correct, consistent and resolve possible conflicts between them.
- Documenting requirements: a good documentation will facilitate its further implementation. Requirements can be documented in various forms such as use cases, specifications, etc.
- Validating requirements: make sure that the system supports all the requirements correctly.

Software requirements give a complete description of the system to be developed. They describe what a system must do and how it must be done. The traditional software engineering distinguishes between two types of requirements:

- Functional requirements: describe the system functionalities.
- Non functional requirements: describe other restrictions of the system (e.g. response time) or about how its development must be (e.g. use a specific programming language)

3.2.1.3.2 Design

Once obtained the requirements, the second phase in the MPIu+a process is the design. Software design is the process of problem planning for a software solution. It includes the planning of algorithms as well as the architecture of the system.

One of the most important parts of interactive systems is the dialog between the user and the system. The user interface is the part of the system that allows the user to interact with it. The user interface determines the user's perception and conception of the application [61].

The interaction design is divided into two activities:

- Activity design: the activity design includes the analysis of functionalities and tasks that users can carry out.
- Information design: related with the interface layout, its components and their style.

An important aspect in this phase is the affordance. The affordance or intuitive comprehension [55] is the ability of an element of the user interface to give the impression that it can be correctly used.

3.2.1.3.3 Implementation

Usability Engineering and the MPIu+a process model do not specifically describe this phase since it corresponds to the classic software development process. The final product is created in this phase. Developers must select the programming languages, database systems and the best technologies that fit the system.

3.2.1.3.4 Launch

Software launch or release is one of the most critical phases in the development process. The product's success will depend on two important factors:

- The degree to which the user is comfortable with the system: the system errors, its simplicity, its functionalities, etc.
- The degree to which the project responsables obtain the expected results.

The MPIu+a process model helps these factors to be satisfied since the design has been done focusing on users and for users. They have been involved in all the process.

3.2.1.4 Prototyping

Prototypes are documents, designs or systems are incomplete versions of a software program being developed. They simulate or implement parts of the final system [62], that can be different from the final product.

Prototyping has several benefits: the final users can get involved in the development process and the software designers can get feedback from them. With prototypes it is possible to evaluate the product from the beginning of the development. Prototyping can also be used to obtain requirements that were not considered previously [63].

Nielsen describes two dimension of prototypes [56]:

- **Horizontal prototypes:** they provide a broad view of the entire system and its interface but they don't implement much functionalities. They focus on the user interaction more than in the system's functionality. An horizontal prototype is a simulation of the interface in which no real tasks can be done [64].
- **Vertical prototypes:** they provide a more complete elaboration of a part of the system. A vertical prototype can prove a limited part of the system but under real circumstances.

Prototyping techniques can be classified according to the fidelity with which they resemble the actual product in terms of appearance, interaction and timing [65]:

- **Low fidelity:** they implement general features of the system without going into details. They are economical, easy to build and they don't require the use of any specific tools.
- **High fidelity:** they represent more precise features of the system. They can detail specific tasks or features. They are more expensive, they require more time and the use of specific tools.

Some of the prototyping techniques are: sketching, storyboards, paper prototypes, navigational storyboards, software prototypes, etc.

3.2.1.5 Evaluation

Evaluate consists in proving something to know if it works correctly, if it covers the expectations or just to see how it works. Usability evaluation is a major aspect in any UCD methodology. Usability evaluation covers all the methodologies and techniques to analyze the usability and accessibility of a product.

In the MPIu+a process model, the evaluation phase is the key to obtain usable and accessible systems. Evaluation must not be considered as a single step in the development process, but should be practiced throughout the whole lifecycle.

According to DIX [66], evaluation has three main objectives:

- Check the system functionalities.
- Check the user interface and its effect effect on users.
- Identify any other specific problem related with the system.

Usability evaluation methods can be classified in three categories:

- **Inspection methods:** inspection is the generic name for a set of methods where an expert evaluator inspects a user interface. Inspection methods have different objectives but all them are based on experts' opinions and reports [67]. Two examples of inspection methods are heuristic evaluation [68] and cognitive walkthrough [69].
- **Inquiry methods:** they involve collecting qualitative data from users. These methods require observing and interviewing users to know their opinions, needs or complains about the system. Inquiry methods include interviews, focus groups, questionnaires, etc.
- **Test methods:** in test evaluation methods end users perform concrete tasks using the system or a prototype. Sessions are usually recorded on video. Evaluators collect the most quantitative data such as task completion time, task completion rate, etc. The think aloud protocol [70] is often used to know participants' thoughts on the application while executing tasks.

3.2.2 RITE method

The Rapid Iterative Testing and Evaluation method (RITE) [71] is an iterative usability method. Documented by researchers at Microsoft, it proposes a variation of traditional usability testing performing short iterations. Evaluations differ from traditional ones mainly in the sense that much smaller groups of users are recruited for the tests. However, tests are performed much more frequently and it advocates that changes to the user interface are made as soon as a problem is identified and a solution is clear.

Consequently, results for individual test iterations are less significant from a statistical and quantitative point of view. The main results from a testing session are basically qualitative and are used to guide the next development iteration. However, as many evaluation iterations are accumulated along the development process, it is possible to perform a quantitative analysis of the results. Moreover, the overall costs of the evaluation are significantly reduced.

Chapter 4

Contribution

4.1 Iteration 1

4.1.1 Requirements analysis

4.1.1.1 Defining end users

One of the requirements of the Semantic Web is to be usable by both tech-savvy users and non-technical users. The complexity of Linked Data limits its use to those who can read and understand how RDF and other Semantic Web technologies like SPARQL work [72, 45].

Different types of users can have different requirements in the use of the Semantic Web. Therefore, its is necessary to identify the main target user groups expected to use the Web of Data. Different user profiles have different skills, requirements and they carry out different tasks [73]. In this project we define mainly three types of users:

- **Tech-users:** users with experience in software but also in Semantic Web technologies, who understand RDF as a data format and are able to interpret an ontological model.
- **Lay-users:** users who do not know Semantic Web technologies, RDF or ontology models. This kind of users are able to find information in Internet through resources such as search engines or Wikipedia.
- **Domain-expert users:** this kind of users may not necessarily have knowledge of software technologies but have an expert knowledge of a concrete domain. They are likely to have a good understanding of the data structure and contents, that allows them to interact with large amounts of complex and heterogeneous data.

4.1.1.2 Requirements

The initial requirements of this project are related with the tasks for data analysis proposed by Shneiderman [51] A brainstorming session was also performed by some of the GRIHO members in order to get ideas and confirm the identified requirements. The RITE methodology and the evaluations performed also helped to refine them.

4.1.1.2.1 Functional requirements

- **Data overview:** users must be able to get global overviews of the data, useful to understand the data structure and find out what the data is about.
- **Data exploration:** users must be able to explore data through visual filters. They don't need to have knowledge of semantic web technologies.
- **History of actions:** users must be able to see the history of actions and go back to previous pages.
- **Visual presentation:** it is necessary to identify suitable methods for presenting the data to all potential end users, hiding the complexity of RDF.

4.1.1.2.2 Non-functional requirements

- **Support for scalability:** the IA components must be able to manage large amounts of complex and heterogeneous data.
- **Non-domain specific:** the IA components must compatible with multiple domains.
- **Standards:** the system should conform to established Semantic Web standards.
- **Browser compatibility:** the IA components must be compatible with the main web browsers.
- **Open source:** the source code must be published using a free license.

4.1.2 Design

4.1.2.1 Navigation menus

Overview is the first user task when dealing with a dataset. The objective is that the user is capable of getting an idea about the overall structure of the dataset. They can be used as starting point for navigation. However, overviews become difficult to achieve with large heterogeneous datasets, which is typical for Linked Data. A common approach to obtain an overview and support the exploration of large datasets is to structure them hierarchically [74]. Hierarchies allow users to visualize different abstractions of the underlying data at different levels of detail

In the case of Semantic Web and Linked Data dataset, this overview is usually about which are the main kinds of things in the dataset, the more instantiated classes, and how they are structured, their hierarchical structure. It is also possible to obtain an overview from the point of view of the more common subjects the data is about and how they are structured, for instance as thesaurus.

Navigation menus, in the case of website, let users navigate through different sections and pages of the site. They tend to be the only consistent navigation element, being present on every page of the site.

Traditionally, user-centred design techniques are used to develop the navigation menus of a site. The typical one is Card Sorting [75], where users are given a set of cards labelled with the main topics of the site and they group these cards following their own criteria. In order to generate menus as meaningful as possible for the broader range of users, the card sorting is repeated with different users. This technique requires a lot of time and effort from developers and most of this effort is wasted as soon as the structure of the menu is established and fixed in a menu that becomes something static. If new kinds of items are introduced or a part of the content becomes more relevant, the Card Sorting should be repeated, at least in part.

In the case of web sites build on top of semantic data, we have the opportunity to automate part of the process of generation and maintenance of the navigation menus. This is possible because semantic data is structured by thesaurus and ontologies, that hierarchically organize the kinds of things described in the dataset. They specify all the classes or categories but also which subjects belong to each class or category.

The objective is to generate a global navigation menu that takes into account all the classes considered in a dataset but also how they are instantiated. Consequently, if there are few instances of some classes or they are not instantiated at all, they should be less relevant in the menu bar. On the contrary, classes that do have a lot of instances should

be shown prominently in the menu bar. This way the menu facilitates the access to the more significant classes but also makes it possible for new users to realise what are the main kinds of things in a dataset.

This approach makes it possible to show the user the navigation bar that best fits the data in the dataset at that particular moment. For instance, if the dataset changes from containing mainly data about projects to mainly about publications, the menu would change accordingly to show more prominently the part of the dataset structure about publications.

On the other hand, one possible drawback of this approach, as it has been pointed by some usability expert evaluations [76], is that users find it very disturbing that the navigation menus change from visit to visit due to changes in the underlying data. This is an inconvenient effect of navigation menus dynamism, as users see them as a static part of the site and, as they get used to them, they rely on them as a handful guide to the site.

In any case, our experiments show that these changes are only perceivable for small datasets. Under those circumstances, the navigation menu undergoes changes quite often when adding new resources. However, as more resources are introduced, changes in the navigation menu tend to be minimal. As soon as the amount of data is statistically significant to keep the natural tendency in the dataset evolution, the changes in the menu bar are practically inexistent or not significant from the point of view of the user. They only affect to particular options in the submenus that are added or removed in the context of more general options in the menu, that keep users in the track to the information they need.

4.1.2.2 Facets

Users don't always know exactly what they are looking for and, sometimes, they don't even know how it is named or the terms used to describe it. Other times, they are unfamiliar with the domain or they want to learn about a topic. This is particularly true when facing Semantic Web datasets. In these cases, exploratory search [77] is a strategy that allows users to refine their search by successive iterations. An exploratory interface such as faceted browsing allows users to find information without a priori knowledge of its schema. This is commonly found in the exploration of RDF datasets, where the users need to identify classes and properties from the schema and learn about the domain.

With navigation menus we can make the user aware of the hierarchical structure of a dataset but, once they choose the class of things they are interested in, the face the barrier of not knowing how they are described. In other words, what are the main properties that describe them, which ones are the more relevant for that particular kind of things, the range of values they have in that particular case, etc.

Faceted navigation is an exploratory technique for navigating a collection of elements in multiple ways, rather than a single and pre-determined order [78, 79, 80]. Facet browser interfaces provide a user-friendly way to navigate through a wide range of data collections. A faceted classification system allows contents to be classified in multiple dimensions. These dimensions are called facets and represent characteristics of the information elements. For example, a collection of books can be classified using an author facet, a subject facet, a date facet, etc.

Facet browser interfaces for RDF were originally demonstrated in the `Flamenco System` [81] and have become popular thanks to projects like `MUSEUM FINLAND` [82]. In the Semantic Web, expressed in RDF, resources constitute the collection of browsed elements and facets are the properties that describe them.

Traditional facet browsers rely on manual identification of the facets and on a previous knowledge of the target domain. Facet browsers are developed to navigate through homogeneous data and facets are fixed. This conflicts with Semantic Web, where data is too diverse to use a single set of facets: facets that make sense for one type of resource could be inappropriate for other types.

When dealing with semantic data, it is possible to automate this process. However, since the Semantic Web integrates data from lot of sources, we can't assume a single fixed schema for all data. Therefore, a semantic faceted browser should be able to handle any RDF dataset without any configuration. It should be scalable and generic, not depending on a particular dataset. Moreover, when new data is added the system should be able to add new facets at run time.

One of the most important aspects of a facet browser is that, when constraining the dataset, all properties and values that would lead to an empty set of results need to be automatically removed from the interface, protecting the user against dead ends. Facets and values need to be changed on the fly. This will save user's time.

4.1.2.3 Breadcrumbs

Breadcrumbs are navigation components used in user interfaces to keep a track of user's location or history of actions within a website [83]. Their name comes from the trail of breadcrumbs left by Hansel and Gretel in the fairytale. Breadcrumbs usually appear horizontally below menu bars or headers and they provide a trail for the user to follow back to the starting point.

Breadcrumbs can reinforce the idea that users are in the right place. If users are disoriented [84, 85] they can select one of the previous breadcrumb links to go back to a previous and known point. Then, they can keep with their goal.

Although the breadcrumb metaphor means to mark the specific path the user has taken, there are different kinds of breadcrumbs and some of them do not extrictly follow this metaphor. It is possible to distinguish between three types of breadcrumbs [86] that can be applied in the semantic web context:

4.1.2.3.1 Location breadcrumbs

Location breadcrumbs are always static and show where the page is located in the website hierarchy. In the context of the semantic web, they indicate the position of a resource in the class hierarchy (or other possible hierarchies such as SKOS concepts hierarchy). They are the simplest and most used because they are easy to implement. Location breadcrumbs are static because a concrete resource has always the same breadcrumb, no matter how users get there.

4.1.2.3.2 Path breadcrumbs

Path breadcrumbs are dynamic and show the path the user has taken to arrive at that page. They are the more adequate representation of the breadcrumb metaphor. They indicate how the user got to the current resource and they show the previous resources the user visited before. A concrete resource can have different breadcrumb paths because users can take different routes to get there. They are useful for websites with graph-like structure, which is the case of the semantic web.

4.1.2.3.3 Attribute breadcrumbs

Attribute breadcrumbs give meta-information that categorizes the current page. They represent the classification of a resource showing what categories it belongs to. A concrete resource can have many attribute breadcrumbs, representing its different possible classifications. Like with location breadcrumbs, users can have several possible paths to reach a resource depending on the properties they use to filter.

4.1.3 Prototyping

Figure 4.1 shows a paper prototype with the Information Architecture components proposed in the Approach section. This prototype was not used for user tests, but to place the components in the user interface.

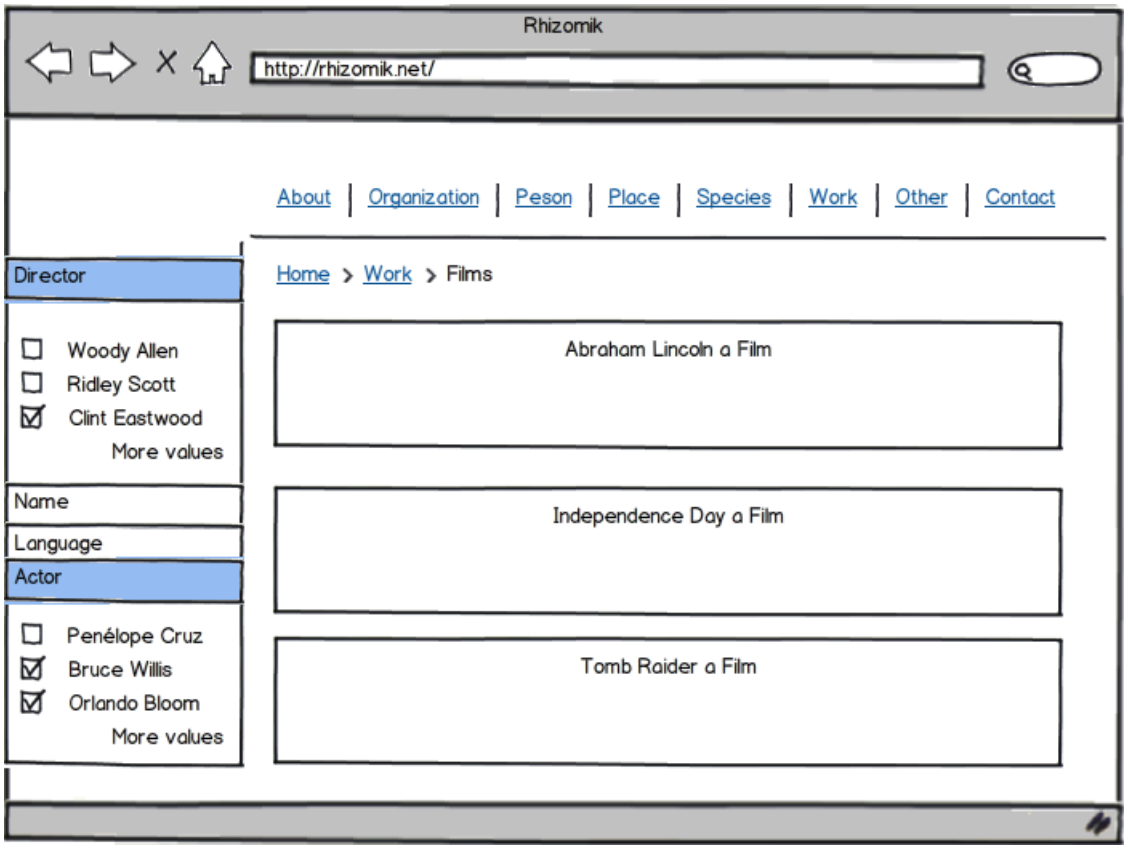

Figure 4.1: Paper prototype

Figure 4.2 shows the software prototype implemented and available at `http://rhizomik.net/linkedmdb`. This prototype was used for the evaluation with end users.

4.1.4 Implementation

4.1.4.1 Navigation menus

The navigation menus are implemented using the Jena Ontology API [87] by obtaining a hierarchical list of domain classes and applying inference rules to get new relations between them. This list of classes is stored in a data structure and then used to generate the navigation menu. For each class, the structure stores information about the number of

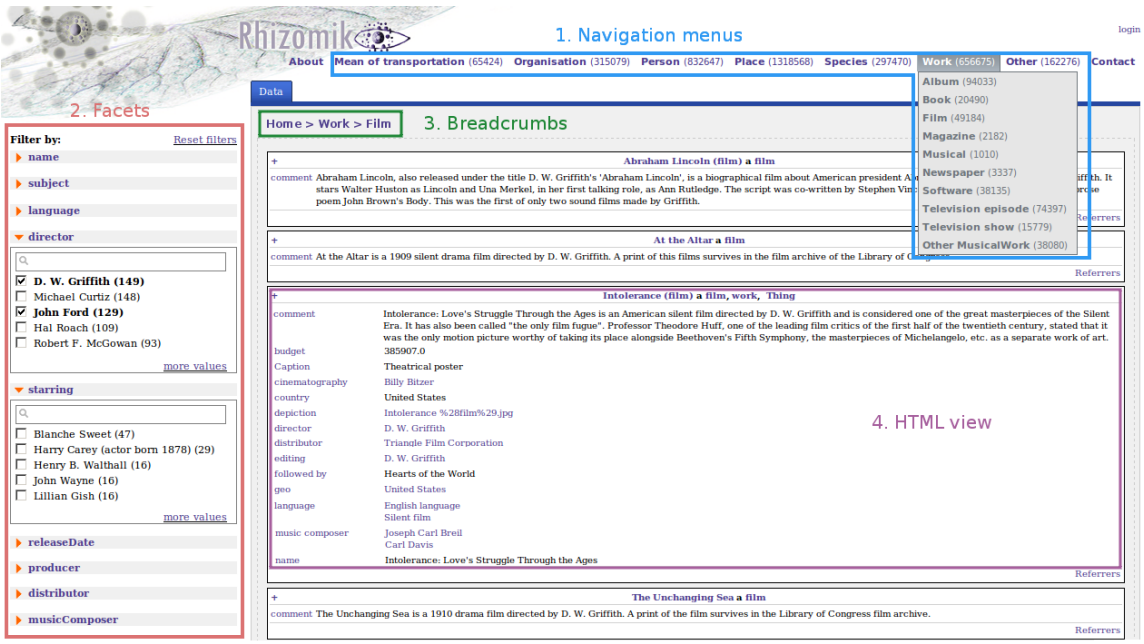

Figure 4.2: Software prototype

instances of the class, its URI, its labels and a list of its subclasses. Using this information it is possible to generate a hierarchical menu that represents all the classes of the domain. All this information is retrieved using the following SPARQL queries:

```
SELECT DISTINCT ?root
WHERE {
  ?root rdf:type ?class .
  FILTER (?class=owl:Class || ?class=rdfs:Class)
  OPTIONAL {
    ?root rdfs:subClassOf ?super .
    FILTER (?root!=?super && ?super!=owl:Thing
      &&?super!=rdfs:Resource && !isBlank(?super))
  }
  FILTER (!bound(?super) && isURI(?root) &&
    !isBlank(?root) && ?root!=owl:Thing )
}
```

Example 3: SPARQL query to get root classes

```
SELECT DISTINCT ?sub
WHERE {
  ?sub rdfs:subClassOf %classURI%
  OPTIONAL {
    ?sub rdfs:subClassOf ?sub2 .
    ?sub2 rdfs:subClassOf %classURI% .
    FILTER (?sub!=?sub2 && ?sub2!=<%1$s>
         && !isBlank(?sub2))
  }
  FILTER (!bound(?sub2))
}
```

Example 4: SPARQL query to get direct subclasses for the given class

```
SELECT ?class COUNT(?x)
WHERE {
  ?x a ?class
}
GROUP BY ?class
```

Example 5: SPARQL query to get the number of instances of each instantiated class

Example 3 shows the SPARQL query to obtain the root classes, i.e. those non-blank without a superclass different from `owl:Thing` or `rdfs:Resource`. Example 4 shows the SPARQL query to get direct subclasses for a given class, i.e. there is not an intermediary class between them in the hierarchy. Finally, Example 5 shows the SPARQL query to get the number of instances of each instantiated class.

A similar approach can be used to generate a navigation menu with SKOS concept hierarchies [88].

This component can generate both global and local menus, i.e. a menu for the whole dataset or for a subset of it. The site administrator can also configure some parameters:

- The number of levels in the hierarchical menu.
- The number of items in each level of the menu.
- The order of items: alphabetically or by number of instances.
- A list of classes or namespaces to omit.

According to these parameters, this component generates the menu applying a recursive algorithm, shown in Example 6, that mainly performs two operations:

- Split those classes with a large amount of instances in subclasses.
- Group those classes with few instances in a superclass.

```
generateMenu(Menu menu, int numItems) {
  menu.removeEmpty();
  while(menu.size()>numItems) {
    Node other = menu.createOther();
    Node min = menu.getMinNode();
    other.mergeWith(min);
  }
  while(menu.size()<numItems){
    Node max = menu.getMaxNode();
    menu.splitNode(max);
  }
}
```

Example 6: Overview of the navigation menu generation algorithm

The algorithm starts with a menu tree structure that initially contains the whole hierarchy of classes and the number of instances for each class. The first step of the algorithm is to remove all the empty classes that have zero instances. Then, depending on the number of intended items in the final menu, i.e. parameter "numItems", the algorithm performs mainly two operations:

- If the number of menu items is higher than the input parameter, those classes with fewer instances are grouped in a new class called "Other".
- If the number of menu items is smaller than the input parameter, the class with more instances is divided into its subclasses.

These operations are recursively performed until the menu is completed. Figure 4.3 illustrates the process of generating the navigation menu for a subset of DBPedia, with 7 elements in the first level. In the original hierarchy there are only 3 classes in the first level. Therefore, there are 4 free spots in the menu. To cover these free spots, the algorithm identifies which classes are appropriate to divide, taking into account their number of instances and their number of subclasses.

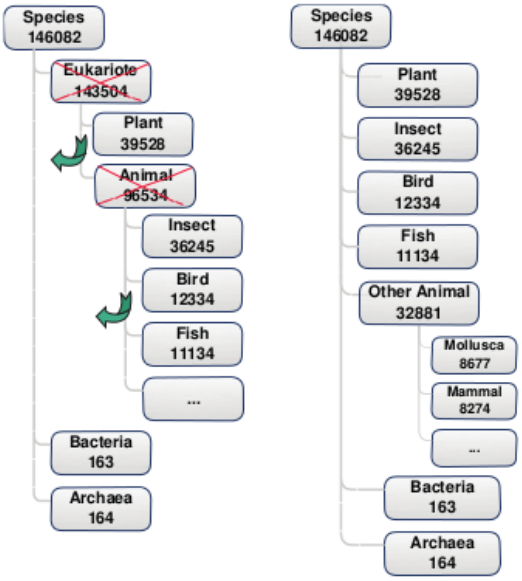

Figure 4.3: Generating a navigation submenu for DBPedia Species with 7 slots (left original, right result)

At first, the `Eukariote` class is removed and its subclasses, `Plant` and `Animal`, move up to a higher level in the hierarchy. After this step, the navigation menu contains 4 elements: `Plant`, `Animal`, `Bacteria` and `Archaea`. From here, the algorithm is applied recursively until the menu is completely generated. In the next step, the `Animal` class is chosen and divided. However, in this case, there is not space for all its subclasses in the first level of the menu. For this reason, the subclasses with a higher number of instances move up to the main level of the menu while the rest of subclasses are grouped inside `Other Animal`.

It is important to note that the procedure depicted so far takes into account the whole dataset classes and instances at a given moment and generates the corresponding menu as an static snapshot. Figure 4.4 shows the complete menu generated for the DBpedia dataset.

4.1.4.2 Facets

To build the facets, and to keep them updated, what Rhizomer does is to perform SPARQL queries for each class in the dataset that retrieve all the properties their instances have, the different values for each property and the cardinality for each value, i.e. how many times that property for that class takes that value.

Facets are pre-calculated and stored in a data structure. They are updated when the user starts browsing and selecting values for different facets. In this case, the set of instances used for facets generation is constrained by the choices made so far and the facets are recalculated for that constrained set of instances. Those facets that are no longer relevant, i.e. no instance uses them, are removed from the facets set. For instance, if the value "2010" has been selected for facet "date", and for that date there is no instance in the dataset with the property "completedOn", then this facet will not be included in the set shown to the user.

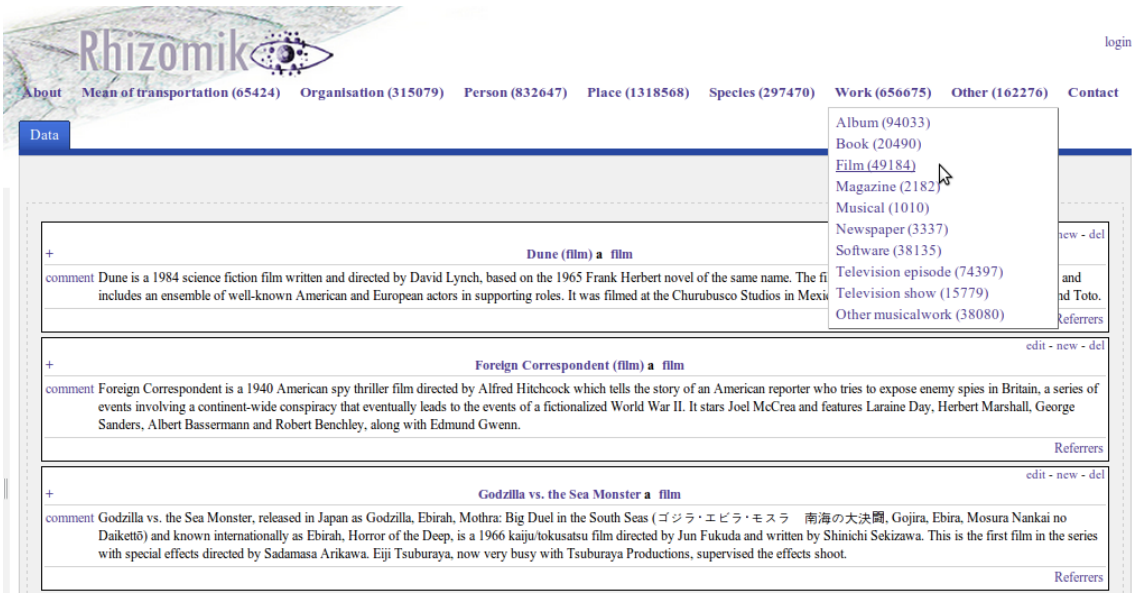

Figure 4.4: Navigation menu generated for the DBPedia

4.1.4.2.1 Automatic facet discovery and ranking

When a dataset is very large and heterogeneous, the number of facets will also be very large. Therefore, it is needed an automated method to choose which facets are more useful and important for the user. We need to find those facets that best represent the dataset and those that are best to navigate the dataset. Choosing the right facets is very important. A suitable facet should allow efficient navigation through the dataset and be representative for those objects.

Faceted browsing can be seen as a decision tree. A path in the tree represents a set of constraints that select the resources of interest. As the tree is constructed dynamically and the information space changes, facets need to be recalculated at each step of the decision tree. To measure the quality of a facet, and therefore showing it more prominently to the user, we use three metrics:

- **Predicate frequency:** we are interested in those predicates that occur frequently inside the instances being browsed. The more resources covered by the predicate, the more useful it is in dividing the information space. If the predicate is not frequent it will only affect a small subset of the collection. We compute the predicate frequency as the number of resources for which the predicate p is defined. We normalise this value dividing it by the total number of resources:

$$freq(p) = \frac{n_r(p)}{n_r}$$

- **Predicate balance:** the facet helps the user better discriminate the set of instances being browsed when it takes a well-balanced range of values for the facet property. On the contrary, a facet whose property takes always or mainly a particular value is less useful. The same happens if each instance has a different value for the facet property. Consequently, we will favour facets that show behaviours in between these worst cases. To compute the predicate balance we use the Shannon's entropy formula:

$$H(S) = -\sum_{i=1}^{n} p(v_i) \log_n p(v_i)$$

- **Value cardinality:** a suitable predicate should have a small amount of values to choose from. If there are too many choices it is difficult to display all the options and it might confuse the user. We compute the value cardinality as the number of different values for a predicate. This metric is normalized using a function based on the Gaussian density that can be regulated through the μ and σ parameters to the top and bottom values of the range of different values we are interested in. This range is still to be fixed experimentally but existing work recommends ranges similar to from 2 to 20 [28]:

$$card(p) = \begin{cases} 0 & \text{if } n_o(p) \leq 1 \\ e^{-\frac{(n_0(p)-\mu)^2}{2\sigma^2}} & \text{otherwise} \end{cases}$$

The three metrics are combined using a function with equal weights that produces a unique usefulness value for each facet. Previous works [28] suggest that facet metrics should be recalculated at each step. However, once facets have been generated and prioritised given their usefulness, we keep their order in the user interface because too many changes in the UI could cause usability problems.

Examples 7, 8, 9 and 10 show the queries used to retrieve all the information necessary to build facets and calculate the previous metrics. These queries are performed for each class and are stored in a local sqlite[1] database. These queries also retrieve labels when they are available:

```
SELECT DISTINCT ?p ?r
WHERE {
  ?x a <CLASS> ; ?p ?o
  OPTIONAL { ?p rdfs:range ?r }
  FILTER (?o != "")
  FILTER (?p!=owl:differentFrom && ?p!=owl:sameAs)
}
```

Example 7: SPARQL query to obtain all properties for a <CLASS>

```
SELECT ?o (COUNT(?o) AS ?n) ?label
WHERE {
  ?r a <CLASS>; <PROPERTY> ?o .
  OPTIONAL{ ?o rdfs:label ?label }
}
GROUP BY ?o ?label
ORDER BY DESC(?n)
```

Example 8: SPARQL query to obtain values and counts for a <CLASS> and <PROPERTY>

```
SELECT (COUNT(?x) AS ?n)
WHERE {
?x a <CLASS> ; <PROPERTY> ?o
}
```

Example 9: SPARQL query to obtain the number of instances of a <CLASS> that have a concrete <PROPERTY>

[1] http://www.sqlite.org/

```
SELECT (COUNT(DISTINCT(?o)) AS ?n)
WHERE {
  ?x a <CLASS> ; <PROPERTY> ?o .
  FILTER (?o!="")
}
```

Example 10: SPARQL query to obtain the cardinality for a concrete <CLASS> and <PROPERTY>

These pre-computed facets are used whenever a user wants to browse a class, usually selecting it from the navigation bar. When a user starts interacting with them by selecting values, facets are dynamically recalculated starting from the pre-computed ones. This makes it possible to provide reasonable response times to users because most of the computing effort has already been done. The user interaction only restricts the set of resources to work with. The set of instances used for facets generation is constrained by the choices made so far and the facets are recalculated for that subset.

Facets are rendered as HTML, as it shown in Figure 4.5. The interface shows a sidebar with the target facets selected using the metrics described before. Each facet consists in a list with the five most used values and a text search box, which suggests possible matches. There is also the possibility to see the rest of values and choose from them.

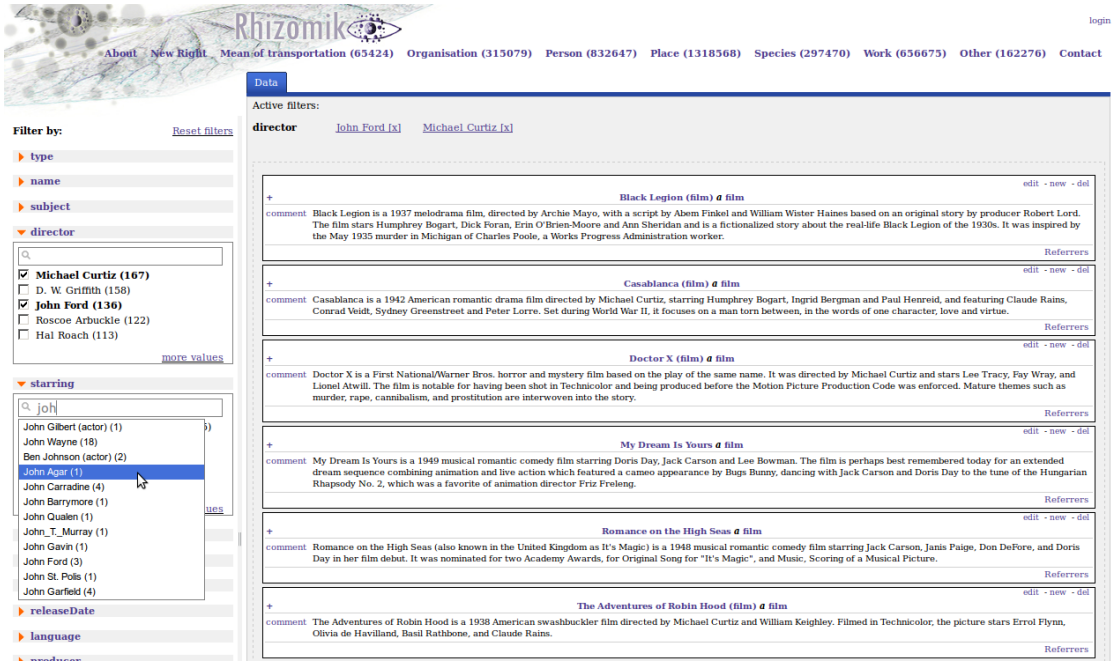

Figure 4.5: Automatic facets for the `http://dbpedia.org/ontology/Film` class

4.1.4.3 Breadcrumbs

The three types of breadcrumbs presented can be implemented for the semantic web. In this iteration we have only implemented attribute breadcrumbs because they are essential for facets. The other types of breadcrumbs will be considered in further iterations.

In faceted search interfaces, attribute breadcrumbs show selected facet values and allow users to remove the selected filters. Users must always know where they are in the faceted browser. It is important to show which facets they have chosen and they must be able to remove the selected filters.

To implement breadcrumbs we store a collection of key-value pairs. For each selected property we store its uri as the key and a list of the selected values for that property. Breadcrumbs are displayed horizontally with each selected property in a different line.

4.1.5 Evaluation

The developed IA components were tested with end-users in order to evaluate its functionality and usability. The goal of the test conducted so far was to do a preliminary evaluation of the Information Architecture components, if they are understood and if they improve the awareness of the structure of a particular dataset by improving user performance when looking for a specific piece of information.

The tests were performed in the UsabiliLAB[2] facilities at Universitat de Lleida. To register sessions we used Morae Recorder and Morae Observer to analyse user test data. We used a real test dataset called the Linked Movie Database (LinkedMDB) [3]. We chose the movies domain because it is well know for most people and quite appealing. LinkedMDB is generated from the Internet Movie Database3 (IMDB), data is extracted from the IMDB site, represented as Linked Data and enriched with an ontology. LinkedMDB is generated from the Internet Movie Database (IMDB)[4], data is extracted from the IMDB site, represented as Linked Data and enriched with an ontology. We chose the movies domain because it is well know for most people and quite appealing.

4.1.5.1 Objectives

The objectives of the evaluation are:

- Compare the navigation systems offered by Rhizomer with LinkedMDB and IMDB.com.
- Evaluate the Information Architecture components developed and the overall interaction of the Rhizomer platform.
- Obtain spontaneous feedback from the users and detect other usability problems that were not previously considered.

4.1.5.2 User profiles

Six participants were selected, with a unique profile characterized by good knowledge of information technology, limited knowledge about Semantic Web technologies and interest in movies.

4.1.5.3 Methodology

The participants were recruited and filled out a pre-test form. These information was necessary to determine if they belong to the desired user profile. Then, they signed a confidentiality document, giving permission to be recorded.

The test had three phases:

[2] `http://griho.udl.cat/en/infraestructures/usabililab.html`

[3] LinkedMDB by O. Hassanzadeh and M. Consens, awarded 1st prize at the Linked Open Data Triplification Challenge 2008, `http://triplify.org/Challenge/2008`

[4] `http://www.imdb.com/`

- Free exploration of the prototype: the participant gets in touch with the system and explores it.
- Task realization: the participant receives a document with instructions for each tasks. He must perform the task and describe his interaction with the system. After performing each task, the participant must fill out a form about it.
- Interview and comments: after completing all the tasks, the participant is interviewed and he can give comments about the system and the tasks. He must also fill out a post-test form.

During all the evaluation the think-aloud protocol [70] was used. This method used in usability tests proposes that participants express their thoughts on the application while performing test tasks.

4.1.5.4 Tasks

We considered interesting to compare the evaluation results with those for IMDB and thus be able to test if the same data as Linked Data can become more usable than from the original web site. Consequently, we established one scenario with one task to be performed with IMDB and another one with one task for Rhizomer.

The test facilitator proposed users the two scenarios and tasks, but not necessarily in the same order:

- **Task 1:** "Find three films where Woody Allen is director and actor at the same time" using IMdB.
- **Task 2:** "Find three films where Clint Eastwood is director and actor at the same time" using Rhizomer.

4.1.5.5 Usability metrics

For the usability test we chose the following metrics:

- Effectiveness: defined as the percentage of tasks completed.
- Efficiency: defined as the percentage of completed tasks per time.
- Time per task.
- Number of times asking the facilitator for help.

We didn't consider user satisfaction yet because the focus at the current stage is on the previous usability metrics. This metric will be considered in the next iterations.

4.1.5.6 Results

The main findings from the test are presented in Table 4.1 and analysed next:

- Only one participant was able to complete the first task without assistance.
- 100% of participants needed in at least one occasion the guidance of the facilitator to successfully complete the second task.

Task	Efficacy	Efficiency	Time (m)	Help[5]
Task 1 (IMDB)	100%	32%	3.37	5
Task 2 (Rhizomer)	100%	54%	2.41	6

Table 4.1: Evaluation results for the first iteration

- In task 2, 100% of the participants began the navigation from actors instead than from movies. This was the reason why users required assistance but as soon as they realized they were able to start it from movies, the tasks was easily solved.
- Both tasks required a lot of interaction steps to be completed. The first task was completed with an average of 8.67 steps and 9.83 in the second task.
- Efficiency is relatively low for both tasks. 32% on average in the first task, and 54% in the second task.
- 83% of participants completed the second task in less time than the first. Just one user completed the first task in less time than the second.

From test results and their analysis, these proposals were elaborated to improve the IA components developed and the Rhizomer platform:

- Navigation must be better contextualised. The interface should provide more mechanisms to inform the user where he is, where he can go and where he has been. For that, the proposal is to integrate breadcrumbs in natural language that summarise the navigation steps though navigation menus and facets.
- All items should be labelled so URIs or URI fragments are not shown to the user. For resources that have no label, this requires a tool that detects unlabelled items and creates a label for them automatically.
- A pagination mechanism is necessary to make it clear the total number of results and to allow browsing them.
- Improve how facets are presented to the user, especially when there are a lot of facets and a lot of values for a concrete facet. For that, the proposal is to use values indexes or graphical representations for numeric values, e.g. histograms.
- Mark the external links, using some sort of image, text or colour, so in case the user leaves the application he is aware in advance.
- Hide some advanced features, like data edition, that are not useful for non-advanced users. Different user profiles will be defined and we will determine which options are displayed to each user profile.
- The main issue detected is that the user interaction is currently too constrained by how the underlying data is structured. In this test, the task with Rhizomer was performed differently from how it was expected and this confused all users. They were looking for movies where actor and director were the same. Instead of initiating their interaction from the `Movies` menu option, all users started from `Actor`. From there, as the underlying data just modelled actors per film but not the reverse, it was impossible to filter those films where the same person was the director. The easy way

was to look for movies and to filter by director and actor using the corresponding facets, as the underlying data has these two properties associated to every film. The impression is that users tend to think first about persons and consider films a secondary entity. The idea here is to exploit the possibilities of the underlying conceptual model and derive implicit properties, for instance reverse properties, in order to provide users with alternative paths. In this particular case, there will be reverse properties from actors to films. Moreover, it will be necessary then to focus on the set of films for an actor and filter it by director.

4.1.5.7 Conclusions

The first tests with users show that Rhizomer facilitates publishing and browsing a dataset, like many other similar tools, but also allows users to realise what is the value of the dataset in the context of their particular needs. This is accomplished by the developed information architecture components. It has also shown the scalability of these components, from small datasets to really big ones like LinkedMDB.

The tests also show that users can have a better performance using Rhizomer with a Linked Data dataset than with the original user interface. However, the user interaction is currently too constrained by how the underlying data is structured. The idea here is to exploit the possibilities of the underlying conceptual model and offer alternative navigation paths. In this particular case, deriving reverse properties from actors to films could improve the user interaction.

4.2 Iteration 2

4.2.1 Requirements analysis

The evaluation performed in the first iteration showed mainly that users need more ways to interact with data and a better contextualization. When analysing the evaluation results, it became evident that the fact users started from actors was the reason why they required assistance. They arrived at a dead-end after filtering actors by name to just "Woody Allen" and there was no way to switch to his set of films and then filter it using the director facet.

A short path to this problem might be to add for each resource, e.g. "Woody Allen (Actor)", a link for each facet to the set of resources that can be reached through it, e.g. a link to all the films where Woody Allen has acted. However, this just works for particular instances and the objective is to make it also work for sets of resources, e.g. all the films by a Spanish actor.

Another solution to this problem is to add to each class faceted-view some derived facets, i.e. facets from other classes that are directly connected to the current one through a property. For instance, add the "directed by" facet to actors derived from the "director" facet of the films they have acted in. However, this approach does not scale well because the number of facets for each class gets easily unmanageable and derived facets quite easily lose their context and become confusing. For instance, it can be difficult to distinguish between the "country" facet for author birthplace and a "country" facet derived from the country of the films the actor has participated in.

This motivates the development of a pivot operation to switch between different types of resources. We also obtained the following formal requirements for this iteration:

- **Pivoting support:** it is necessary a mechanism that allows to switch between different types in facet browsing, e.g. from actors to films.
- **Better contextualization:** attribute breadcrumbs in facet browsing are not clear enough. It is necessary to improve the contextualization and show users where they are and what are they seeing in facets.
- **Automatic generation of labels:** some resources don't have a label. In this cases, an automatic label must be generated using the resource's URI.

4.2.2 Design

Pivoting in facets

Pivoting is not a common feature of existing Linked Data exploration tools. As reviewed in the State of the Art section, the only active tools capable of providing a faceted view and pivoting on semantic data are Parallax[6] and Virtuoso Faceted Browser[7]. However, Parallax is constrained to the Freebase dataset and facets and pivoting in Virtuoso Faceted Browser are not evident and are quite difficult to use.

From the point of view of OLAP systems[89], pivoting or rotation is described as an operation producing a change in the dimensional orientation of data. For instance, if data is initially aggregated by Product, Location and Date, by pivoting, the user can aggregate, for instance, by Location, Date and Product.

[6] `http://www.freebase.com/labs/parallax/`

[7] `http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtuosoFacetsWebService`

For richer data models such as Linked Data, pivot navigation is a way to restart a search from the results of a first search[90]. Usually, the type of resources to be browsed (e.g. film, actor, director...) remains fixed in a faceted browsing application. However, when pivoting is added to faceted navigation, it allows switching the type of displayed entities based on relations to the current result set. For instance, a user who is filtering films using film facets, e.g. director is "Woody Allen", then pivots on actors. As a result of this action, the user will see now all actors in the result list, who are related to any film in the previous filtered list. Then, the user can continue filtering but now using actor facets, e.g. country is "Spain".

It is possible to establish an analogy between pivoting and natural language. Indeed, the query above can be rephrased as "Show actors from Spain, which have acted in films directed by Woody Allen". The idea of pivot is reflected by the fact that the set of "Spanish actors" in the main sentence also appears in the relative sentence as the relative pronoun "which". The relative pronoun points to the facet to browse for a pivot, in this case "acted in".

Pivot steps can be repeated, e.g. pivot on countries from actors and filter continent is "Europe", after removing the previous country is "Spain" filter from actors. Each pivot step corresponds to a nested relative sentences, such as "Show European countries, where an actor, which has acted in a Woody Allen film, has been born".

Literal breadcrumbs

The pivoting operation allows users to make more complex queries between different types of resources. However, breadcrumbs become also more complex and difficult for users to understand. The resemblance between pivoting and natural language can be used to generate more usable breadcrumbs that help users contextualise their exploration. This way, they know why they are getting the list of results that they are looking at as a result of their filtering and they see the pivoting steps so far.

Look-ahead breadcrumbs

Regular breadcrumbs show the trail of links leading to the current resource. Look-ahead breadcrumbs [91] can help the navigation by including a list of links to resources that are reachable from that particular resource. A study of look-ahead breadcrumbs suggest that they can lead to more efficient site navigation and improve the user experience [92].

Look-ahead breadcrumbs can be used to show a list of classes for which it is possible to pivot to from the current faceted view. This way, users can easily identify possible navigation paths and the pivot operation looks like following common links to other pages.

Labels

Applications for linked data consuming are intended to be widely used by different kinds of users. Therefore, hiding technical details such as URIs when displaying data to users becomes crucial. Entities in the Semantic Web need to have labels in order to be exposble to humans in a meaningful way. Labels are used for displaying the entities when exploring the data instead of displaying the URIs. They can can also be used to support keyword-based or natural-language-based search. The property `rdfs:label` is usually used to provide a human-readable version of the resource's name besides its URI[8].

4.2.3 Implementation

Pivoting in facets

The first step to implement pivot-enabled facets is to determine which ones should provide pivot. Properties with XML Schema data type or RDF Literal values, for a given class, result in facets that do not provide pivoting. On the other hand, properties that connect to other resources allow pivoting. To build facets that support pivoting, we first distinguish three types of properties:

- Datatype properties: properties whose values are RDF literals or data types from XML Schema. It is not possible to pivot on these properties but recognising them allows displaying them with specialised facet types, e.g. a histogram facet for numbers or a calendar one for dates. These specialised facets will be considered in further iterations.
- Object properties: properties whose values reference other resources. These properties were treated, prior to the introduction of pivoting, as facets with literal values, where the values were resources labels. It continues to be possible to filter a set of resources based on the labels of the referenced resources, e.g. filter films through the actor facet based on the actors' labels. However, pivots makes also possible to switch to the set of actors and perform a more detailed filtering based on actors' facets.
- Inverse properties: in some cases, a dataset has a property between resource types modelled just in one way. For instance, each resource of type Film has the property actor, but the resources of type Actor do not have the inverse property to relate them with the films they have appeared in. When inverse properties are not explicit in a dataset, they are detected and facets are generated following the same approach than for explicit object properties. Consequently, it is possible to pivot through explicit object properties and also through implicit inverse object properties. This increases the flexibility of the exploration as more choices are available to the user dead-ends are avoided, like exploring actors and not being able to pivot to films because the property from actors to films is not explicitly stated in the dataset.

Those properties which reference to other subjects (object properties) or are inverse properties allow pivoting. In this case, it should be determined to which class the facet links. The faceted view for that class will become the new view when pivoting is performed. This distinction is made by analysing the underlying dataset and ontologies. It results in an additional facet characteristic: its range. The procedure to determine a facet range is the following:

1. Check if, for the given class and property, there is an OWL restriction that defines the property range. This range is selected as the facet range. It can be either a class, XML Schema data type or RDF literal.
2. If no restriction is found in the previous step, it is checked if the property has a defined range, which becomes the property range.
3. If there is no property range, the dataset is analysed and the 5 most common values (Figure 11) for the class and property are retrieved. They are checked to determine whether they are resources or not.

(a) If all the 5 values are resources, then the dataset is queried to determine the most instantiated classes by the values of the property. At most five of them are retrieved using the query shown in Example 11. This list of common classes is then passed to the query in the second row of Example 12. This query retrieves the most specific superclass of all the input classes. The result is then considered the range of the facet and that class will become the new faceted view when the user pivots the facet.

(b) If not all 5 values are resources, their data type, if present, is retrieved or computed by trying to parse their values as an integer, double or date. By default, the value is considered to be a string. Then, the range of the facet is set to the most specific super datatype in the XML Schema datatypes hierarchy[8]. As no pivoting is enabled for this kind of facets, the range might be used to create specific facets for numeric values (like histograms with range selectors) or calendars.

```
SELECT ?type (COUNT(?type) AS ?n)
WHERE {
  ?x a %classURI% ; %propertyURI% ?o .
  ?o a ?type .
}
GROUP BY ?type
ORDER BY DESC(?n)
LIMIT 5
```

Example 11: SPARQL query that retrieves at most the 5 most common classes instantiated by the values of a facet

```
SELECT DISTINCT ?common
WHERE {
  ?common ^rdfs:subClassOf %listOf5CommonClasses%
  OPTIONAL {
    ?intermediate ^rdfs:subClassOf %listOf5CommonClasses%
    ?intermediate rdfs:subClassOf ?common.
    FILTER (?intermediate!=?common && !isBlank(?intermediate)) }
  FILTER (!BOUND(?intermediate) && !isBlank(?common))
}
```

Example 12: SPARQL query that computes the most specific common superclass

Rhizomer keeps track of all pivoting operations and records the initial class, the pivot property and the target class. Example 13 shows the SPARQL query generated when browsing films whose director is "Woody Allen".

```
SELECT DISTINCT ?r1 WHERE {
?r1 a <http://data.linkedmdb.org/resource/movie/film> .
?r1 <http://data.linkedmdb.org/resource/movie/director>
<http://data.linkedmdb.org/resource/director/8501> }
```

Example 13: Generated SPARQL query before pivoting

When pivoting to the new class, the restrictions applied to the previous ones are propagated to the pivoted class through the property used for pivoting and SPARQL variables. For example, when pivoting from films to actors:

- Initial type: `http://data.linkedmdb.org/resource/movie/film`
- Pivoting property: `http://data.linkedmdb.org/resource/movie/actor`
- Pivoting type: `http://data.linkedmdb.org/resource/movie/actor`

[8] `http://www.w3.org/TR/xmlschema11-2/#built-in-datatypes`

The constraints capturing pivoting switch are introduced in the generated query, as shown in Example 14 in lines 3-4. A new variable `r2` is introduced together with its type, i.e. the range of the originating facet. The link from the previous variable `r1` to the new one is established using the pivoted property, i.e. `movie:actor`. Finally, the selected variable is the new variable as the focus has changed from films to actors.

```
1 SELECT DISTINCT ?r2
2 WHERE {
3   ?r2 a movie:actor .
4   ?r1 movie:actor ?r2 .
5   ?r1 a movie:film .
6   ?r1 movie:director <http://data.linkedmdb.org/resource/director/8501>
7 }
```

Example 14: Generated SPARQL query after pivoting

Once the facets that should provide pivoting are determined, this option is offered to users as part of the facet using an arrow shaped link. Those facets that allow pivoting are also listed as part of the look-ahead breadcrumbs. Figure 4.6 shows the pivoting enhancements: pivot-able facets with an arrow icon, breadcrumbs as natural language rendering of the query and look-ahead breadcrumbs with pivoting destinations on the right.

Figure 4.6: Pivoting enhancements

Labels

If a resource doesn't have the `rdfs:label` property, the last part of the URI is used to deal with this problem [93]:

1. If the URI contains a local name, the last part of the URI is used. For example, for the URI `http://dbpedia.org/resource/Berlin` the last part of the path is used, i.e. Berlin.

2. If the URI contains a fragment identifier, the last part of the URI is used. For example, for the URI `http://www.example.com/about#Bob` the last part of the path is used, i.e. Bob.

4.2.4 Evaluation

The aim of the test was mainly to validate that the introduction of pivoting solves the problems highlighted in the previous evaluation. Therefore, we followed the same

methodology and we chose the same user profile. Six users that were not involved in the previous evaluation rounds were recruited.

4.2.4.1 Tasks

Since the main objective of the test was to validate if there was improvement with pivoting, we considered interesting to keep one task from the first evaluation. Task 2, was identical to one used in the previous evaluation round. It was used to test if pivoting had improved efficiency and efficacy. The complete set of tasks was:

- **Task 1:** find 5 films with "Orlando Bloom" as actor.
- **Task 2:** find 5 films with "Clint Eastwood" both as director and actor.
- **Task 3:** find who has directed more films in countries located in "Oceania".

4.2.4.2 Usability metrics

For the usability test we chose the following metrics:

- Effectiveness: defined as the percentage of tasks completed.
- Efficiency: defined as the percentage of completed tasks per time.
- Time per task.
- Number of times asking the facilitator for help.

4.2.4.3 Results

The efficiency results, i.e. time to complete the task, are shown in Table 4.2. The table shows the results for Task 2 with pivoting, prior to pivoting and the observed improvement. Efficiency for Tasks 1 and 2 is also presented.

Time on Task (minutes)	Task 2 with pivoting	Task 2 pre-pivoting	Improvement	Task 1 with pivoting	Task 3 with pivoting
Minimum	0.89	1.05	15%	1.00	1.99
Maximum	2.23	5.23	57%	4.53	4.50
Mean	1.69	2.41	30%	1.61	3.43
Standard Dev.	0.57	1.49	62%	1.21	0.96

Table 4.2: Evaluation results for the second iteration: Efficiency (time on task)

The first finding has been that the introduction of pivoting corresponded to a great increase of efficiency, with a 30% reduction in the mean time necessary to complete Task 2. However, the biggest improvement has been in the reduction of the maximum time on tasks, with 57% improvement.

From the point of view of effectiveness, it is important to note that all users completed the task without facilitator help, while in the previous iteration for the same task all users required facilitator assistance.

This is related with the fact that, thanks to pivoting, all users where able to find their path to solve the task without requiring assistance. Contrary to pre-pivoting tests, where

most users got lost when trying to complete the tasks starting from actor or director instead of from film, with pivoting all users were able to complete the task independently of their starting point without assistance. Consequently, the maximum time was reduced significantly.

However, the following issues were detected and some proposals to solve them are presented to be considered in further iterations of the development

- It was difficult for users to identify the pivoting button. Moreover, the box labelled "Navigate to", that also contained the list of facets that provided pivoting, was far from the facets and hard to identify. Finally, some users thought that following one of this pivoting links meant starting the exploration from the target class from zero, loosing all the restrictions applied so far through facets.

- Users also experienced many contextualisation problems, not being completely obvious for them what was presented to them at the screen. The breadcrumbs helped solving this once the users realised they were available. However, it took some time for most of the users to understand this.

4.2.4.4 Conclusions

The issues detected are mainly related to the fact that the interface components providing pivoting are not so evident for users. Moreover, they suppose a conceptual shift that should be mitigated. For instance, some users understood pivoting as restarting the exploration for a new class of resources.

One possible way to overcome these limitations of pivoting is to try to integrate it with facets, so that users do not need to move their attention from them. It is also necessary to make it clearer that the filtering done so far is not going to be lost. One way to achieve this is to present pivoting as a way of performing some advanced filtering on a facet. This way, users can start doing "classical" filtering using the labels of the facet values. For instance, filter the actors for a film using the actor facet and the actor labels. However, if they get stuck because they need a more sophisticated filter, we can make pivoting available as a way of attaining advanced filtering.

Another issue is related with contextualisation. Though natural language-inspired breadcrumbs have been seen by users as very useful, they should get more prominent in the user interface because it took too much time to users to spot them. The idea in this case is to increase breadcrumbs size and use colours that highlight the breadcrumbs and also the entities involved, i.e. the names of the classes and facets involved. A similar approach should be done with look-ahead breadcrumbs in order to show them in a more prominently in the user interface.

Chapter 5

Conclusion

5.1 Conclusions

The Web has evolved and nowadays the Semantic Web has grown from a vision to a reality. The amount of semantic data available in the Web is rapidly increasing, especially thanks to Linked Data principles and best practices, which have been adopted by an increasing number of data providers. However, end-users find it difficult to explore and use this data.

After some rounds of development and testing with end-users, it is possible to conclude that the main hypotheses of this work were correctly posed. Rhizomer is capable of publishing semantic data while facilitating user awareness of what information is contained in the dataset. Awareness is accomplished by components borrowed from the Information Architecture discipline and generated automatically from the dataset structure and ontologies.

These IA components are able to use the semantics captured by ontologies and semantic data, providing users different ways to access and interact with the data. Currently, these components are navigation bars, which show the main kinds of items in the dataset, facets, which show the more significant properties for different kinds of items and their values, and breadcrumbs, which allow users to go back to previous pages. These components, which are automatically generated, facilitate publishing and browsing a dataset without requiring a priori knowledge of it or experience in Semantic Web tools.

The evaluation with users also showed the system's scalability from small datasets to really big ones like the whole DBPedia[1] or LinkedMDB[2].

5.2 Future work

The future work can be classified in three lines. First, to implement the improvements detected during the user tests and related with the user interface and the developed IA components. Second, to identify and implement new IA systems to support the tasks for data analysis. And third, to improve the system's performance.

Improvements detected

The main issues detected are mainly related to the fact that the interface components providing pivoting are not so evident for users. They suppose a conceptual shift that should be mitigated. Although natural breadcrumbs improved the contextualization, the users sometimes still get lost. We plan to integrate a SPARQL to Natural Language tool[3] to improve them.

For facets, one objective is to generate facets customised to the kind of values being managed, i.e. numerical values, alphabetical values, dates, geographical points, etc. Another objective is to improve the functionality of this component by adding other operators for restricting the results: inverse selection, existential selection, join selection, selection between ranges, etc.

Once implemented these improvements, a new user evaluation should be performed. Besides effectiveness and efficiency, the user satisfaction must be also considered. It is also necessary to perform user tests with other user profiles in order to obtain new feedback about the user interface and identify other usability problems. Another possibility is to evaluate user interaction with eye tracking techniques.

[1] `http://rhizomik.net/dbpedia`
[2] `http://rhizomik.net/linkedmdb`
[3] `http://sparql2nl.aksw.org`

New Information Architecture components

Research has shown [94] that keyword search and facet browsing complement each other. Both interaction styles should be supported simultaneously and a method for combining them is needed. The proposal is to implement multi-type facet browsing that can be applied to a set of results from a keyword search.

We also plan to develop prototypes for the other identified IA components, test and evaluate them with users. For instance, *Treemaps* [95] or *CropCircles* [96] can be used to provide high level overviews of datasets. They are a good method to display the size of each node in a hierarchy.

System's performance

One of the main issues detected is related with the system's performance. We realized that it had a negative impact on the user experience because of the time it took to compute some queries. This issue is of the utmost importance and is being addressed for future versions.

In order to avoid repeating the whole menu and facets generation process each time that changes on the dataset are performed, Rhizomer should monitor all changes to the dataset. Navigation menus and facets should be updated whenever the dataset is edited through Rhizomer. Whenever a change is detected, the records for all the involved classes should be updated accordingly. If a new instance is inserted or removed, all the classes it belongs to should be updated.

Appendix A

User evaluation documents

Consentimiento de participación y grabación

En cumplimiento de la Ley Orgánica 15/1999 de Protección de Datos de Carácter Personal (LOPD), mediante el presente documento confirmo que:

1. Acepto participar en esta prueba de usuario que lleva a cabo el laboratorio de usabilidad UsabiliLAB.
2. Autorizo la filmación en vídeo de la prueba.
3. Esta grabación podrá ser utilizada con finalidades científicas para el análisis de los datos recogidos en el proyecto o para divulgar los resultados, ya sea por parte del GRIHO o por la empresa cliente en presentaciones o reuniones profesionales.
4. Renuncio a los derechos de la grabación de vídeo y entiendo que la grabación se puede utilizar para los fines descritos sin permiso adicional.
5. En ningún caso se podrá hacer un uso que pueda vulnerar mi imagen o dignidad personal ni hacer un uso comercial.
6. Puedo ejercitar los derechos de acceso, rectificación, cancelación y oposición de mis datos personales, de acuerdo a la normativa vigente, comunicándolo a los datos de contacto de los que dispongo.
7. He tomado esta decisión basándome en la información que se me ha proporcionado por escrito y he tenido la oportunidad de recibir información adicional en caso de haberla solicitado.
8. Entiendo que la participación es voluntaria y que puedo retirar este consentimiento en cualquier momento sin recibir una penalización por ello.

Nombre y apellidos del participante:

_____

DNI: _____ Teléfono: _____ Email: _____

Fecha: _____

Firma participante Firma administrador del test

Para más información o para cualquier tema relacionado con el proyecto se puede usted dirigir a:

Josep Maria Brunetti
Universitat de Lleida
josepmbrunetti@diei.udl.cat

Cuestionario Post-tarea

El cuestionario Post-tarea nos va a permitir obtener la información de la tarea de forma inmediata lo que nos permite valorar el grado de percepción de la usabilidad en cada tarea y por ello, en los diferentes apartados del sistema. A continuación se detallan las preguntas.

Fecha de la sesión: ..... / ….. / 2012
Observador/a: .....
Tarea nº:

A continuación le leeré una serie de frases relacionadas con la tarea que acaba de realizar, díganos cuál de ellas se ajusta más a su opinión.

1. La tarea era…

Muy difícil de realizar	1	2	3	4	5	Muy fácil de realizar

2. Pienso que he realizado la tarea …

Nada correctamente	1	2	3	4	5	Muy correctamente

3. La estructura de la interfaz … **…a resolver la tarea.**

No me ha ayudado en absoluto	1	2	3	4	5	Me ha ayudado mucho.

4. La tarea ha resultado …

Muy larga	1	2	3	4	5	Muy corta

5. Para realizar la tarea. ..

He tenido que estar muy concentrado	1	2	3	4	5	No he tenido que estar nada concentrado

6. La tarea ..

No estaba bien definida y fui incapaz de entender qué tenía que hacer	1	2	3	4	5	Estaba bien definida y entendí perfectamente qué tenía que hacer

7. Si usted ha decidido NO completar la tarea, por favor responde a las siguientes cuestiones:

¿Qué ha contribuido al hecho de que no haya podido finalizar la tarea?

8. Comentarios adicionales (ej. sobre la tarea, la interfaz, etc.)

Universidad de Lleida

ID participante: _____

Cuestionario Post-Test

Fecha: _____

	De acuerdo ---------- En desacuerdo				
Preguntas	1	2	3	4	5
Ha sido fácil utilizar esta herramienta.					
El sistema es intuitivo.					
Me he divertido utilizándolo.					
Las opciones disponibles son fácilmente identificables.					

1. ¿Qué es lo que más te ha gustado/disgustado del sistema? ¿Por qué?

_____
_____
_____

2. ¿Cómo te ha parecido la interacción en general con el sistema?

_____
_____
_____

3. ¿Qué cosas cambiarías o mejorarías en esta herramienta?

_____
_____
_____

4. ¿Tienes alguna sugerencia adicional que quisieras compartir?

_____
_____
_____

Bibliography

[1] Tim Berners-lee, Robert Cailliau, and Bernd Pollermann. World-wide web: The information universe. *Communications of the ACM*, 37:76–82, 1992.

[2] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[3] Vladimir Geroimenko and Chaomei Chen, editors. *Visualizing the Semantic Web.* Springer, 2002.

[4] Nigel Shadbolt, Tim Berners-Lee, and Wendy Hall. The semantic web revisited. *IEEE Intelligent Systems*, 21(3):96–101, May 2006.

[5] Tom Heath, John Domingue, and Paul Shabajee. User interaction and uptake challenges to successfully deploying semantic web technologies. In *in Proc. 3rd International Semantic Web User Interaction Workshop*, 2006.

[6] Tim Berners-Lee and Mark Fischetti. *Weaving the Web : The Original Design and Ultimate Destiny of the World Wide Web by its Inventor.* Harper San Francisco, September 1999.

[7] World Wide Web Consortium (W3C). W3c semantic web activity. `http://www.w3.org/2001/sw/`.

[8] Dan Brickley and Ramanathan V. Guha. Rdf vocabulary description language 1.0: Rdf schema. Technical report, 2 2004.

[9] D. Beckett. Rdf/xml syntax specification (revised). Technical report, W3C Recommendation, 2004.

[10] Ramanathan V. Guha and Dan Brickley. RDF Vocabulary Description Language 1.0: RDF Schema. W3C recommendation, W3C, February 2004.

[11] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[12] G. Schreiber and M. Dean. OWL web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, February 2004.

[13] World Wide Web Consortium (W3C). Xml 1.0 specification. `http://www.w3.org/TR/REC-xml/`.

[14] Eric Prud'hommeaux and Andy Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2006.

[15] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(3):1–22, MarMar 2009.

[16] Tim Berners-Lee. Linked data - design issues. *W3C*, (09/20), 2006.

[17] Wolfgang Halb, Alexander Stocker, Harald Mayer, Helmut Mülner, and Ilir Ademi. Towards a commercial adoption of linked open data for online content providers. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 16:1–16:8, New York, NY, USA, 2010. ACM.

[18] Uldis Bojars, Alexandre Passant, Frederick Giasson, and John Breslin. An architecture to discover and query decentralized RDF data. volume 248 of *CEUR Workshop Proceedings ISSN 1613-0073*, June 2007.

[19] Tim Berners-lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[20] T Berners-lee, J. Hollenbach, Kanghao Lu, J. Presbrey, and Mc Schraefel. Tabulator redux: Browsing and writing linked data.

[21] Experimenting with explorator: a direct manipulation generic rdf browser and querying tool. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, February 2009.

[22] Simile: Longwell rdf browser. `http://simile.mit.edu/wiki/Longwell`.

[23] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. /facet: A Browser for Heterogeneous Semantic Web Repositories. In *ISWC*, 2006.

[24] Rasmus Hahn, Christian Bizer, Christopher Sahnwaldt, Christian Herta, Scott Robinson, Michaela Bürgle, Holger Düwiger, and Ulrich Scheel. Faceted wikipedia search. In *BIS*, pages 1–11, 2010.

[25] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *In 6th Intl Semantic Web Conference, Busan, Korea*, pages 11–15. Springer, 2007.

[26] Orri Erling. Faceted views over large-scale linked data.

[27] Marbles. `http://marbles.sourceforge.net/`.

[28] Eyal Oren, Renaud Delbru, and Stefan Decker. Extending faceted navigation for rdf data. In *International Semantic Web Conference*, pages 559–572, 2006.

[29] Rdf gravity. `http://semweb.salzburgresearch.at/apps/rdf-gravity/index.html`.

[30] Isaviz. `http://www.w3.org/2001/11/IsaViz`.

[31] Emmanuel Pietriga. Semantic web data visualization with graph style sheets. In *Proceedings of the 2006 ACM symposium on Software visualization*, SoftVis '06, pages 177–178, New York, NY, USA, 2006. ACM.

[32] Emmanuel Pietriga, Christian Bizer, David Karger, and Ryan Lee. Fresnel - a browser-independent presentation vocabulary for rdf. In *In: Proceedings of the Second International Workshop on Interaction Design and the Semantic Web*, pages 158–171. Springer, 2006.

[33] Tuukka Hastrup, Richard Cyganiak, and Uldis Bojars. Browsing linked data with fenfire, 2008.

[34] Aba-Sah Dadzie and Matthew Rowe. Approaches to visualising linked data: A survey. *Semantic Web*, 2(2):89–124, 2011.

[35] Adapting Graph Visualization, Flavius Frasincar, Ru Telea, and Geert jan Houben. Adapting graph visualization techniques for the visualization of rdf data. In *of RDF data, Visualizing the Semantic Web, 2006*, pages 154–171, 2005.

[36] David Karger and MC Schraefel. The pathetic fallacy of RDF. Position Paper for SWUI06, 2006.

[37] Leonard Richardson and Sam Ruby. *Restful web services*. O'Reilly, first edition, 2007.

[38] Dave Crane, Eric Pascarello, and Darren James. *Ajax in Action*. Manning Publications, October 2005.

[39] Roberto García, Juan Manuel Gimeno, Ferran Perdrix, Rosa Gil, Marta Oliva, Juan Miguel López, Afra Pascual, and Montserrat Sendín. Building a usable and accessible semantic web interaction platform. *World Wide Web*, 13(1-2):143–167, March 2010.

[40] Roberto García, Juan Manuel Gimeno, Ferran Perdrix, Rosa Gil, and Marta Oliva. A platform for object-action semantic web interaction. In *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 404–418, Berlin, Heidelberg, 2008. Springer-Verlag.

[41] Aba-Sah Dadzie, Matthew Rowe, and Daniela Petrelli. Hide the stack: toward usable linked data. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I*, ESWC'11, pages 93–107, Berlin, Heidelberg, 2011. Springer-Verlag.

[42] Riccardo Albertoni, Alessio Bertone, and Monica De Martino. Semantic Web and Information Visualization. In *Semantic Web Applications and Perspectives (SWAP)1st Italian Semantic Web Workshop*, 2004.

[43] Zhenning Shangguan and Deborah L. McGuinness. Towards faceted browsing over linked data. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010.

[44] Ying Ding, Yuyin Sun, Bin Chen, Katy Borner, Li Ding, David Wild, Melanie Wu, Dominic DiFranzo, Alvaro Graves Fuenzalida, Daifeng Li, Stasa Milojevic, ShanShan Chen, Madhuvanthi Sankaranarayanan, and Ioan Toma. Semantic web portal: a platform for better browsing and visualizing semantic data. In *Proceedings of the 6th international conference on Active media technology*, AMT'10, pages 448–460, Berlin, Heidelberg, 2010. Springer-Verlag.

[45] Tom Heath. How will we interact with the web of data? *IEEE Internet Computing*, 12(5):88–91, September 2008.

[46] P. Chandler and J. Sweller. Cognitive load while learning to use a computer program. *Applied Cognitive Psychology.*, 10:151–170, 1996.

[47] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January 2000.

[48] Christian Hirsch, John Hosking, and John Grundy. Interactive visualization tools for exploring the semantic graph of large knowledge spaces.

[49] Lloyd Rutledge, Jacco Van Ossenbruggen, and Lynda Hardman. Making rdf presentable: Integrated global and local semantic web browsing, 2005.

[50] Louis Rosenfeld and Peter Morville. *Information Architecture for the World Wide Web.* O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2nd edition, 2002.

[51] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *IEEE Visual Languages*, number UMCP-CSD CS-TR-3665, pages 336–343, College Park, Maryland 20742, U.S.A., 1996.

[52] Toni Granollers. User centred design process model, integration of usability engineering and software engineering. In *Proceedings of INTERACT 2003*, pages 673–675, 2003.

[53] C Stephanidis. Towards user interfaces for all: Some critical issues. panel session "user interfaces for all-everybody, everywhere, and anytime". In *Symbiosis of Human and Artifact-Future Computing and Design for Human-Computer Interaction, Proceedings of the 6th International Conference on Human-Computer Interaction (HCI International '95*, pages 137–142. Elsevier, Elsevier Science, 1995.

[54] International Standards Organization. *ISO 13407. Human Centred Design Process for Interactive Systems.* Geneva, Swiss, 1999.

[55] D.A. Norman. *The design of everyday things.* New York: Doubleday, 1990.

[56] Jakob Nielsen. *Usability Engineering.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[57] ISO. ISO 9241-11:1998 Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. Technical report, International Organization for Standardization, 1998.

[58] International Organization for Standardization ISO and International Organization for Standardization. *Ergonomics of Human-system Interaction: Guidance on Accessibility for Human-computer Interfaces.* International standard. ISO, 2003.

[59] Web accessibility initiative (wai). `http://www.w3.org/WAI/`.

[60] Ian Sommerville and Gerald Kotonya. *Requirements Engineering: Processes and Techniques.* John Wiley & Sons, Inc., New York, NY, USA, 1998.

[61] Harold Thimbleby. *User Interface Design.* Addison-Wesley, 1990.

[62] Mary Beth Rosson and John M. Carroll. *Usability engineering: scenario-based development of human-computer interaction.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[63] Frederick P. Brooks. *The mythical man-month : essays on software engineering.* Addison-Wesley Pub. Co, anniversary edition, August 1995.

[64] I. Hamilton and A. Life. *Simulation And The User Interface.* Taylor & Francis, 1990.

[65] Marc Rettig. Prototyping for tiny fingers. *Commun. ACM*, 37(4):21–27, April 1994.

[66] Alan Dix, Janet Finlay, Gregory Abowd, and Russell Beale. *Human-computer interaction.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[67] Jakob Nielsen. Usability inspection methods. In *Conference companion on Human factors in computing systems*, CHI '95, pages 377–378, New York, NY, USA, 1995. ACM.

[68] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, CHI '90, pages 249–256, New York, NY, USA, 1990. ACM.

[69] C. Wharton and University of Colorado. Institute of Cognitive Science. *Cognitive Walkthroughs: Instructions, Forms and Examples.* Institute of Cognitive Science, 1992.

[70] C. H. Lewis. Using the "Thinking Aloud" Method In Cognitive Interface Design. Technical Report RC-9265, IBM, 1982.

[71] Michael C. Medlock, Dennis Wixon, Mark Terrano, Ramon L. Romero, and Bill Fulton. Using the RITE method to improve products; a definition and a case study. In *Usability Professionals Association*, 2002.

[72] W. Halb, Y. Raimond, and M. Hausenblas. Building Linked Data For Both Humans and Machines. In *WWW 2008 Workshop: Linked Data on the Web (LDOW2008)*, Beijing, China, 2008.

[73] Ben Shneiderman and Catherine Plaisant. *Designing the User Interface: Strategies for Effective Human-Computer Interaction (4th Edition).* Addison Wesley, 4 edition, April 2004.

[74] Niklas Elmqvist and Jean-Daniel Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Trans. Vis. Comput. Graph.*, 16(3):439–454, 2010.

[75] D. Spencer and J.J. Garrett. *Card Sorting: Designing Usable Categories.* Rosenfeld Media, LLC, 2009.

[76] Roberto García, Juan Manuel Gimeno, Ferran Perdrix, Rosa Gil, Marta Oliva, Juan Miguel López, Afra Pascual, and Montserrat Sendín. Building a usable and accessible semantic web interaction platform. *World Wide Web*, 13(1-2):143–167, March 2010.

[77] Gary Marchionini. Exploratory search: from finding to understanding. *Commun. ACM*, 49(4):41–46, April 2006.

[78] Marti A. Hearst. Uis for faceted navigation recent advances and remaining open problems.

[79] Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Flexible search and navigation using faceted metadata. Technical report, University of Berkeley, 2002.

[80] Marti Hearst, Ame Elliott, Jennifer English, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Finding the flow in web site search. *Commun. ACM*, 45(9):42–49, September 2002.

[81] Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '03, pages 401–408, New York, NY, USA, 2003. ACM.

[82] Eero Hyvönen, Eetu Mäkelä, Mirva Salminen, Arttu Valo, Kim Viljanen, Samppa Saarela, Miikka Junnila, and Suvi Kettula. Museumfinland-finnish museums on the semantic web. *Web Semant.*, 3(2-3):224–241, October 2005.

[83] Mark Bernstein. The bookmark and the compass: orientation tools for hypertext users. *SIGOIS Bull.*, 9(4):34–45, October 1988.

[84] Sougata Mukherjea and James D. Foley. Visualizing the world-wide web with the navigational view builder. *Comput. Netw. ISDN Syst.*, 27(6):1075–1087, April 1995.

[85] Joonah Park and Jinwoo Kim. Effects of contextual navigation aids on browsing diverse web systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '00, pages 257–264, New York, NY, USA, 2000. ACM.

[86] K Instone. Location, path and attribute breadcrumbs. In *3rd Annual Information Architecture Summit*.

[87] Jeremy J. Carroll, Ian Dickinson, Chris Dollin, Dave Reynolds, Andy Seaborne, and Kevin Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 74–83, New York, NY, USA, 2004. ACM.

[88] Alistair Miles and José R. Pérez-Agüera. Skos: Simple knowledge organisation for the web. *Cataloging & Classification Quarterly*, 43(3):69–83, 2007.

[89] E. F. Codd, S. B. Codd, and C. T. Salley. Providing OLAP (On-Line Analytical Processing) to User-Analysis: An IT Mandate, 1993.

[90] G. M. Sacco and S. Ferré. *Dynamic Taxonomies and Faceted Search: Theory, Practice, and Experience*, volume 25 of *The Information Retrieval Series*, chapter 9 - Applications and Experiences. Springer, 2009.

[91] H. Teng. *Location Breadcrumbs for Navigation: an Exploratory Study.* Canadian theses. Thesis (M.C.Sc.)–Dalhousie University, 2004.

[92] James Blustein, Ishtiaq Ahmed, and Keith Instone. An evaluation of look-ahead breadcrumbs for the www. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, HYPERTEXT '05, pages 202–204, New York, NY, USA, 2005. ACM.

[93] Basil Ell, Denny Vrandečic, and Elena Simperl. Labels in the web of data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 162–176, Berlin, Heidelberg, 2011. Springer-Verlag.

[94] Max L. Wilson, Bill Kules, m. c. schraefel, and Ben Shneiderman. From keyword search to exploration: Designing future search interfaces for the web. *Found. Trends Web Sci.*, 2(1):1–97, January 2010.

[95] Ben Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. Graph.*, 11(1):92–99, January 1992.

[96] Taowei David Wang and Bijan Parsia. Cropcircles: Topology sensitive visualization of owl class hierarchies. In *In Proc. of the 5th International Conference on Semantic Web*, pages 695–708, 2006.